User Manual UMAX140600-03
Version 2C
Firmware 2.xx
Axiomatic EA: 5.15.108.0+

USER MANUAL

LIN – J1939 CAN
Protocol Converter

P/N: AX140600-03

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
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. 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 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.[1]

[1] Converters with firmware V1.xx could operate only at 250 kbit/s baud rate.

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.

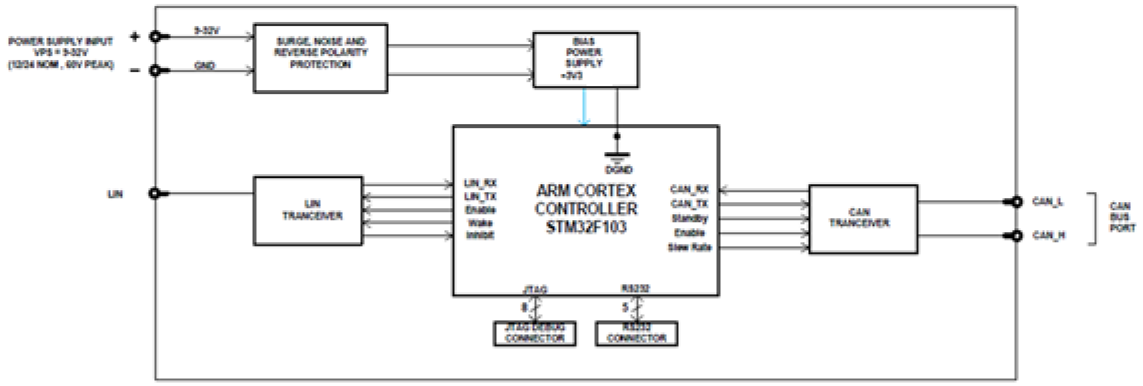


Figure 1. The Converter Hardware Block Diagram

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 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 the *Flashing New Firmware* section for more information on this.

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).
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 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	16 (AX140600-03, 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 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 EA.

2.4.3 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.

2.4.4 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.

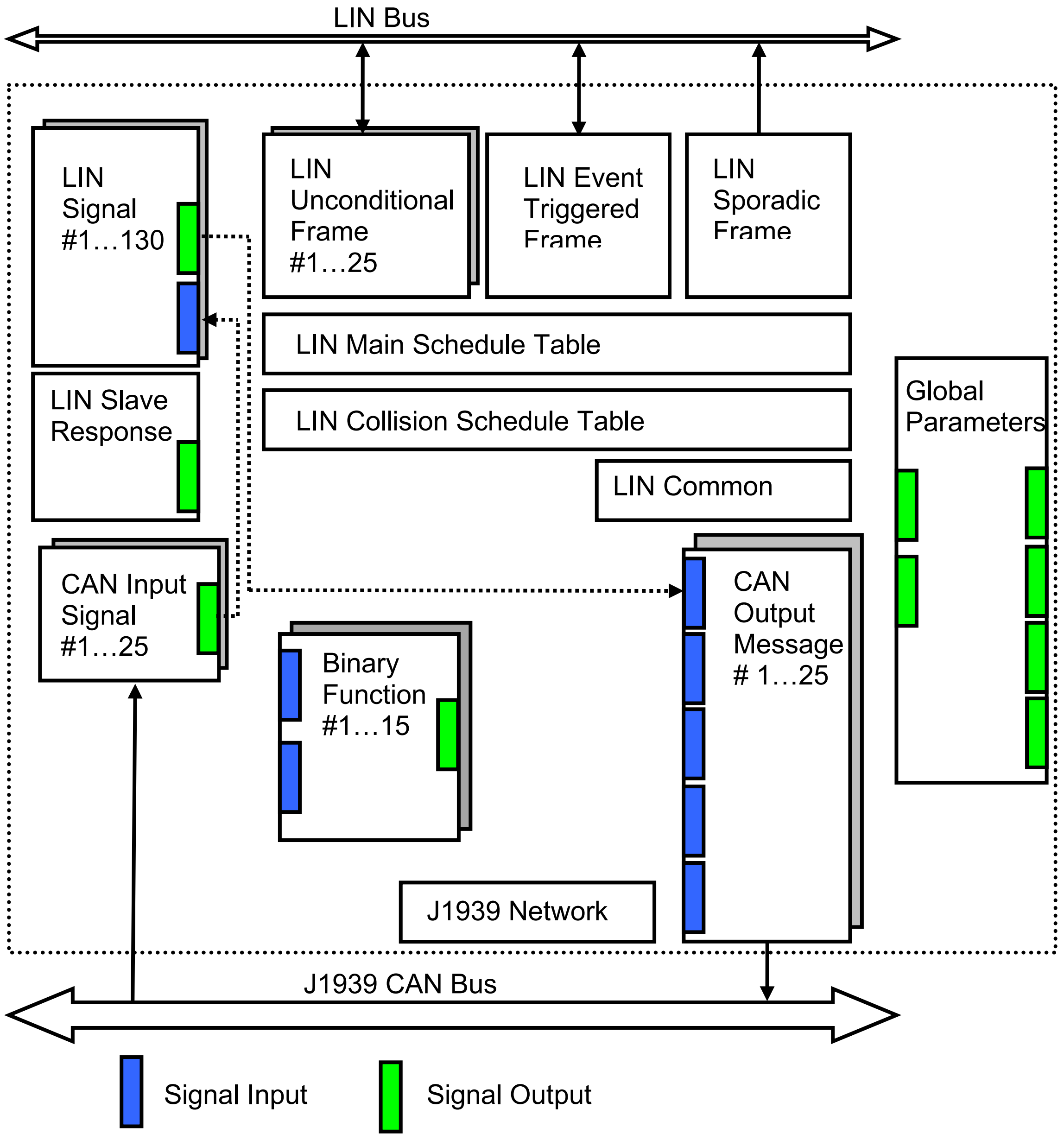
Even if the length of the CAN network is short and the signal reflection from both ends of the cable can be ignored, at least one 120 Ohm resistor is required for the majority of CAN transceivers to operate properly.

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 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 Axiomatic EA.

3.2.1 LIN Common

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

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

Name	Default Value	Range	Units	Description
				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 *130 LIN Signal* function blocks available to the user[1]. Each *LIN Signal* function block has one signal input and one signal output for interfacing with other function blocks.

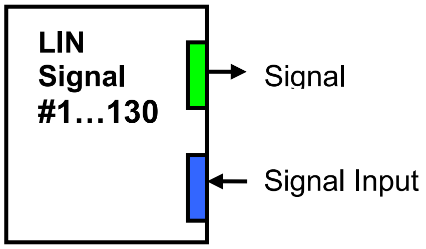


Figure 4. LIN Signal Function Block

[1] *Converters with firmware V1.xx had only 75 LIN Signals.*

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 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”.

Name	Default Value	Range	Units	Description
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:

$$LINOutputSignal = LINSignalCode * Scale + Offset, \quad if \quad LINSignalCode \in [MinValue; MaxValue]. \qquad (1)$$

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.

3.2.2.2 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.

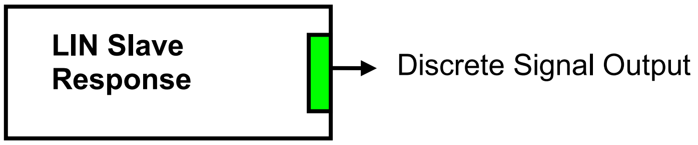


Figure 5. LIN Slave Response Function Block

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 *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.

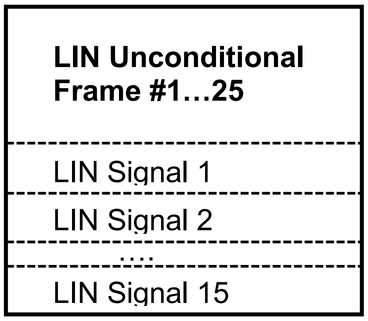


Figure 6. LIN Unconditional Frame Function Block

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 15 LIN signals associated with each LIN frame[1].

[1] *Converters with firmware V1.xx had only 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…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.

3.2.5 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.

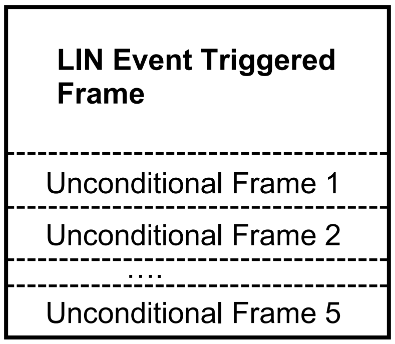


Figure 7. LIN Event Triggered Frame Function Block

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 *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*.

3.2.6 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.

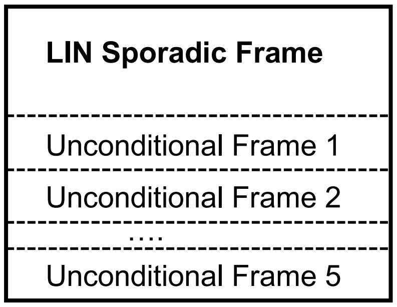


Figure 8. LIN Sporadic Frame Function Block

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…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.

3.2.7 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 25 schedule entries. There are no signal inputs and outputs in this function block.

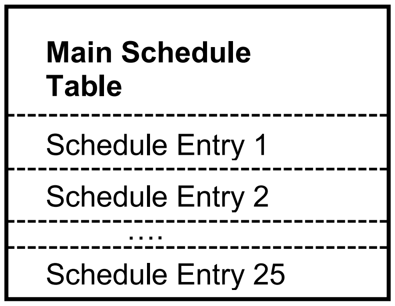


Figure 9. Main Schedule Table Function Block

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 #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.

3.2.8 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.

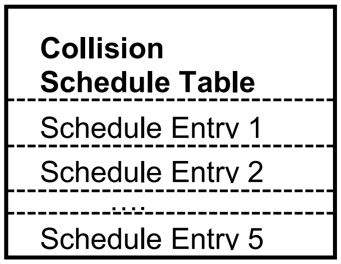


Figure 10. Collision Schedule Table Function Block

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…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.

3.3 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.

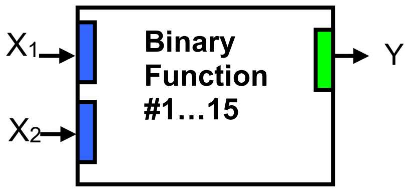


Figure 11. Binary Function Block

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

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

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) = ~x	x is converted into 4-byte unsigned integer before function is applied
abs(x) Absolute	f(x) = x, if x≥0 f(x) = -x, 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] = 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, if x≥y F[x;y] = y, if x<y	
min(x,y) Minimum	F[x;y] = x, if x≤y F[x;y] = y, if x>y	
== Equal	F[x;y] = 1, if x=y F[x;y] = 0, if x≠y	
!= Not Equal	F[x;y] = 1, if x≠y F[x;y] = 0, if x=y	
> Great	F[x;y] = 1, if x>y F[x;y] = 0, if x≤y	
>= Great Equal	F[x;y] = 1, if x≥y	

Function Name	Description	Comment
	$F[x;y] = 0$, if $x<y$	
< Less	$F[x;y] = 1$, if $x<y$ $F[x;y] = 0$, if $x \geq y$	
<= Less Equal	$F[x;y] = 1$, if $x \leq y$ $F[x;y] = 0$, 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	$F[x;y] = x \mid y$	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	$F[x;y] = x \text{ ^ } y$	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
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 *Input #1 Signal Default* is *Yes*.
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 *Input #2 Signal Default* is *Yes*.

Name	Default Value	Range	Units	Description
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* functional block gives the user access to a set of global constants, converter supply voltage and the microcontroller internal temperature.

The function block has one configurable *Global Discrete Constant Signal* output, one configurable *Global Continuous Constant Signal* output and two continuous pre-set constant signal outputs: *Global Const. Signal = 0.0* and *Global Const. Signal = 1.0*.

The function block also contains *Supply Voltage* continuous signal output presenting the converter supply voltage in [V]. Please note, that this voltage is not the voltage on the power supply connector pins. It is an internal voltage measured after the EMI filter, reverse polarity, and transient protection circuit. It is always less than the actual power supply voltage by approximately 0.2…0.5 V.

The microcontroller internal temperature is presented on the *Microcontroller Temperature* continuous signal output in [°C].

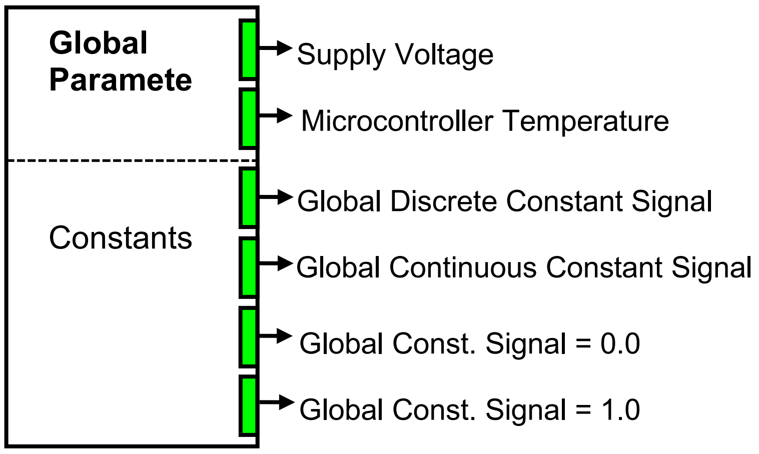

Global Paramete
Supply Voltage
Microcontroller Temperature
Constants
Global Discrete Constant Signal
Global Continuous Constant Signal
Global Const. Signal = 0.0
Global Const. Signal = 1.0

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 Axiomatic 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. They contain *ECU Network* and *CAN Network Parameters*.

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.
Baud Rate[1]	–	{250, 500, 667, 1000}	kbit/s	Current baud rate on the CAN network.
Automatic Baud Rate Detection[2]	Yes	{No, Yes}	–	Set to *No* once ECU is permanently installed on the CAN network.
Slew Rate	Low	{Low, High}	–	Slew rate control of the CAN transceiver.

[1] Read-only parameter. Not available in firmware V1.xx.
[2] Not available in firmware V1.xx.

3.5.1.1 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.

3.5.1.2 CAN Network Parameters

The *Baud Rate* read-only 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.

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

Table 17. Slew Rates

Slew Rate Value	Transceiver Slew Rate	Note
Fast	19 V/µs	Available for all baud rates.
Slow	4 V/ µs	Only available for 250kbit/s baud rate.

The user can select the *Slew Rate* only when the converter operates at 250 kbit/s baud rate. For baud rates higher than 250 kbit/s, the *Slew Rate* is always set to *Fast* independently of the *Slew Rate* configuration parameter.

The *Slow* slew rate is preferable at 250 kbit/s baud rate in the majority of applications due to the reduced EMI of the CAN transceiver. The *Fast* slew rate, in this case, is used when the distance between CAN nodes substantially exceeds 40 m – the maximum value defined by the J1939/11(15) standard.

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.

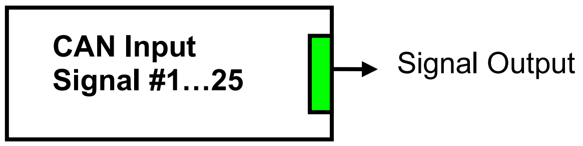


Figure 14. CAN Input Signal Function Block

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.

[1]*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 *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:

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

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

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

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 *25 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 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[1].

[1] *Merging CAN messages is not allowed in firmware V1.xx.*

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

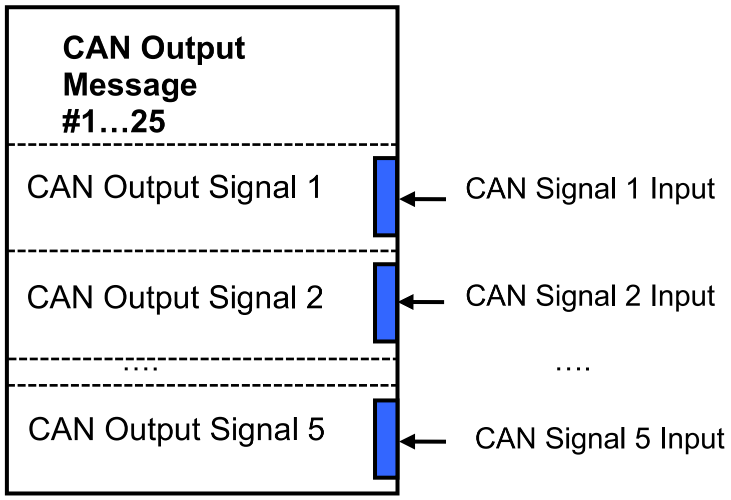


Figure 15. CAN Output Message Function Block

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.
Use CAN Signals in Another CAN Message[2]	No	{Yes, No}	–	Use CAN signals from this message in another CAN message with the same PGN. This CAN message will not be transmitted independently if “Yes”.
Transmission Rate[3]	0	[0;10000]	ms	CAN output message transmission rate. If 0 – transmission is upon request.
Transmit on LIN Unconditional Frame Number[2,3]	0	[0…25]	–	Transmit CAN message on successful transmission or reception of the LIN unconditional frame number.
Destination Address[3]	0xFF	[0; 255]	–	Destination address of the PDU1 PGN messages.
Length[3]	8	[0…8]	–	CAN message data frame length.
Priority[3]	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 #1Offset	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.

Name	Default Value	Range	Units	Description
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 #2Offset	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.
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.

[1]Proprietary A PGN (61184) is excluded. It is taken by Axiomatic Simple Proprietary Protocol and therefore cannot be used in function blocks.
[2]Not available in firmware V1.xx.
[3]Not used if Use CAN Signals in Another CAN Message is set to "Yes". The parameters from the main CAN Output message with the same PGN and with the Use CAN Signals in Another CAN Message configuration parameter set to "No" are used.

When it is necessary to transmit more than 5 CAN output signals in one CAN message, the user defines one main CAN message with the *Use CAN Signals in Another CAN Message* set to *No* and a number of auxiliary CAN messages with the same PGN and the *Use CAN Signals in Another CAN Message* set to *Yes* to define extra CAN output signals for the main CAN message. The user is responsible not to overlap the CAN output signals in the main and auxiliary messages.

The auxiliary CAN messages with the *Use CAN Signals in Another CAN Message* set to *Yes* cannot be transmitted independently and their configuration parameters *Transmission Rate*, *Transmit on LIN Unconditional Frame Number*, *Destination Address*, *Length*, and *Priority* are not used.

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 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 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.

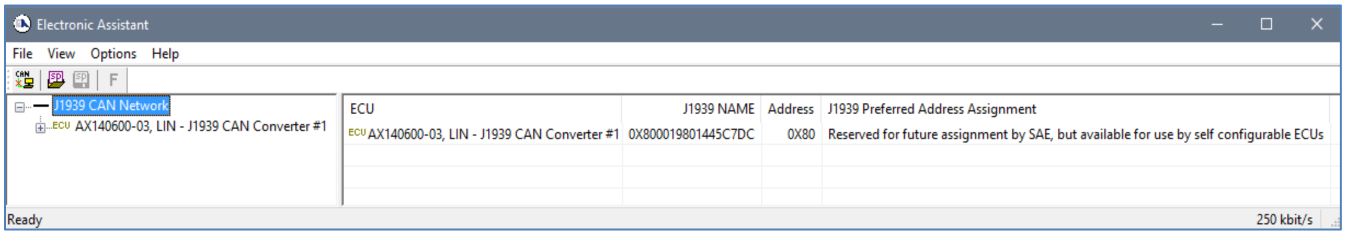
Please, refer to the user manual UMAX07050X for description of the Axiomatic EA and associated products, and for the CAN network connection troubleshooting.

The user should use EA software version 5.15.108.0 or higher, which supports this converter firmware. 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 Figure 16.



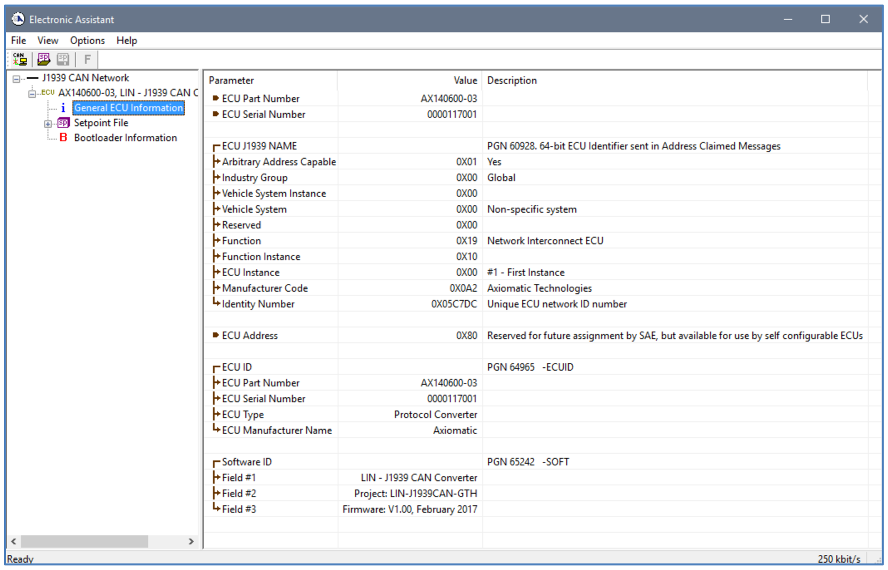
ECU	J1939 NAME	Address	J1939 Preferred Address Assignment
AX140600-03, LIN - J1939 CAN Converter #1	0X800019801445C7DC	0X80	Reserved for future assignment by SAE, but available for use by self configurable ECUs

Figure 16. LIN - J1939 CAN Protocol Converter 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 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.

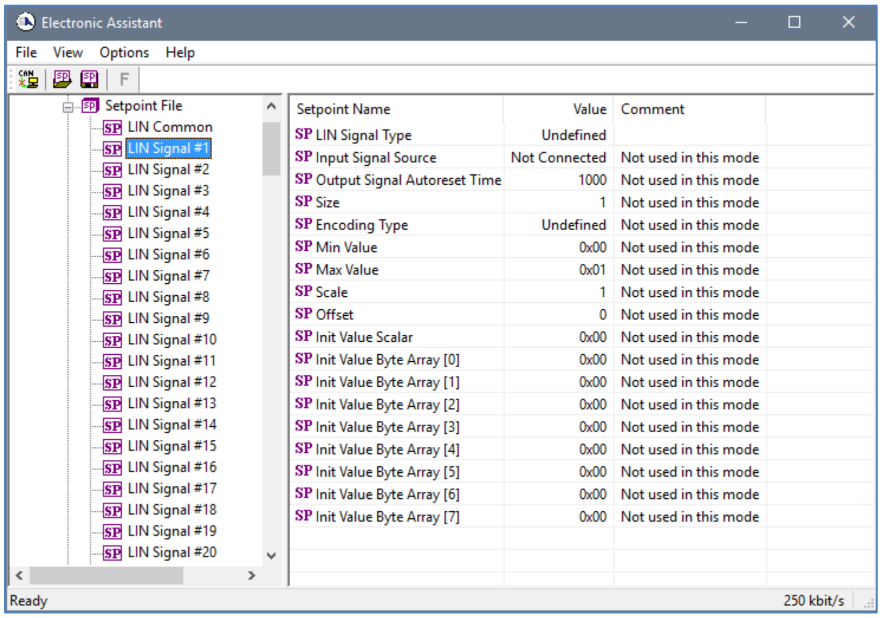


Parameter	Value	Description
ECU Part Number	AX140600-03	
ECU Serial Number	0000117001	
ECU J1939 NAME		PGN 60928. 64-bit ECU Identifier sent in Address Claimed Messages
Arbitrary Address Capable	0X01	Yes
Industry Group	0X00	Global
Vehicle System Instance	0X00	
Vehicle System	0X00	Non-specific system
Reserved	0X00	
Function	0X19	Network Interconnect ECU
Function Instance	0X10	
ECU Instance	0X00	#1 - First Instance
Manufacturer Code	0X0A2	Axiomatic Technologies
Identity Number	0X05C7DC	Unique ECU network ID number
ECU Address	0X80	Reserved for future assignment by SAE, but available for use by self configurable ECUs
ECU ID		PGN 64965 -ECUID
ECU Part Number	AX140600-03	
ECU Serial Number	0000117001	
ECU Type	Protocol Converter	
ECU Manufacturer Name	Axiomatic	
Software ID		PGN 65242 -SOFT
Field #1	LIN - J1939 CAN Converter	
Field #2	Project: LIN-J1939CAN-GTH	
Field #3	Firmware: V1.00, February 2017	

Figure 17. General ECU Information Screen

4.2 Function blocks in the Axiomatic EA

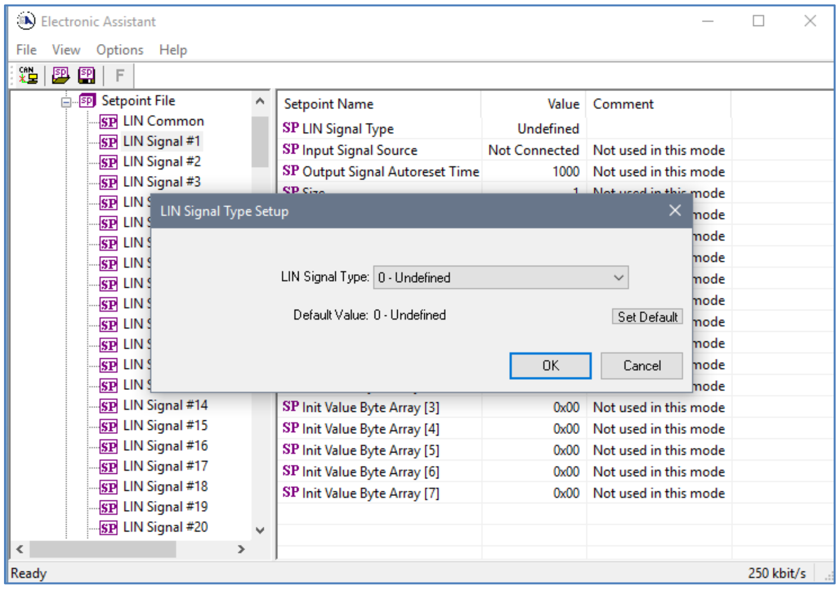
Each converter function block is presented by its own setpoint group in the *Setpoint File* main group. Individual configuration parameters (setpoints) of a function block can be accessed through the function block setpoint group, see Figure 18.



Setpoint Name	Value	Comment
LIN Signal Type	Undefined	
Input Signal Source	Not Connected	Not used in this mode
Output Signal Autoreset Time	1000	Not used in this mode
Size	1	Not used in this mode
Encoding Type	Undefined	Not used in this mode
Min Value	0x00	Not used in this mode
Max Value	0x01	Not used in this mode
Scale	1	Not used in this mode
Offset	0	Not used in this mode
Init Value Scalar	0x00	Not used in this mode
Init Value Byte Array [0]	0x00	Not used in this mode
Init Value Byte Array [1]	0x00	Not used in this mode
Init Value Byte Array [2]	0x00	Not used in this mode
Init Value Byte Array [3]	0x00	Not used in this mode
Init Value Byte Array [4]	0x00	Not used in this mode
Init Value Byte Array [5]	0x00	Not used in this mode
Init Value Byte Array [6]	0x00	Not used in this mode
Init Value Byte Array [7]	0x00	Not used in this mode

Figure 18. 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 19.



Setpoint Name	Value	Comment
SP LIN Signal Type	Undefined	
SP Input Signal Source	Not Connected	Not used in this mode
SP Output Signal Autoreset Time	1000	Not used in this mode
SP Init Value Byte Array [3]	0x00	Not used in this mode
SP Init Value Byte Array [4]	0x00	Not used in this mode
SP Init Value Byte Array [5]	0x00	Not used in this mode
SP Init Value Byte Array [6]	0x00	Not used in this mode
SP Init Value Byte Array [7]	0x00	Not used in this mode

LIN Signal Type Setup

LIN Signal Type: 0 - Undefined

Default Value: 0 - Undefined

Figure 19. 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.

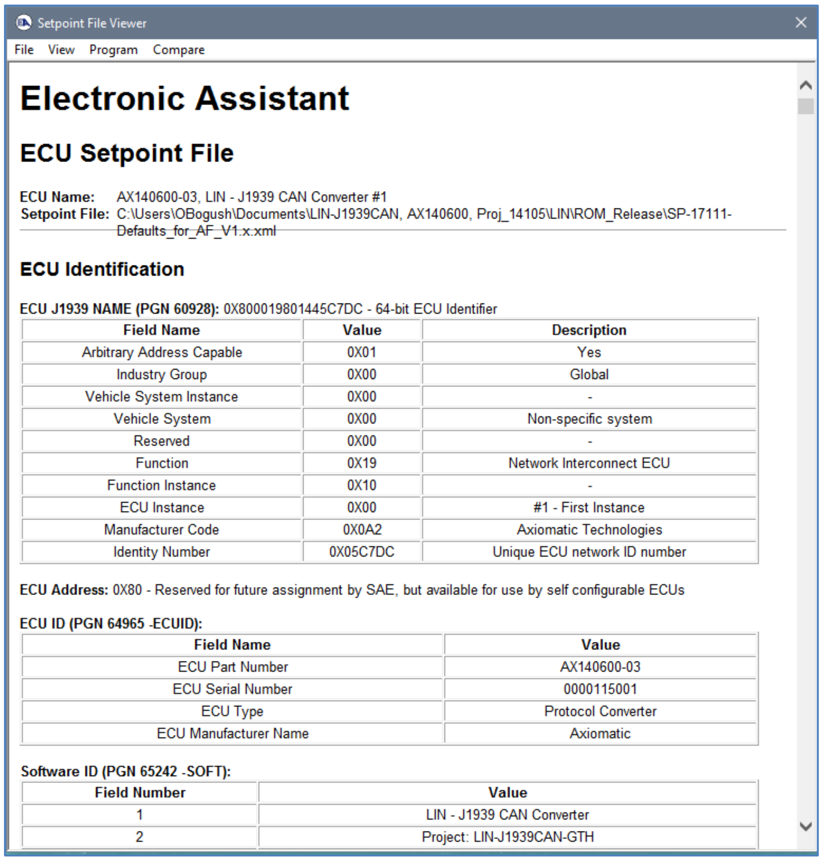
4.3 Setpoint File

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 20.

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.



Electronic Assistant

ECU Setpoint File

ECU Name: AX140600-03, LIN - J1939 CAN Converter #1
Setpoint File: C:\Users\OBogush\Documents\LIN-J1939CAN, AX140600, Proj_14105\LIN\ROM_Release\SP-17111-Defaults_for_AF_V1.x.xml

ECU Identification

ECU J1939 NAME (PGN 60928): 0X800019801445C7DC - 64-bit ECU Identifier

Field Name	Value	Description
Arbitrary Address Capable	0X01	Yes
Industry Group	0X00	Global
Vehicle System Instance	0X00	-
Vehicle System	0X00	Non-specific system
Reserved	0X00	-
Function	0X19	Network Interconnect ECU
Function Instance	0X10	-
ECU Instance	0X00	#1 - First Instance
Manufacturer Code	0X0A2	Axiomatic Technologies
Identity Number	0X05C7DC	Unique ECU network ID number

ECU Address: 0X80 - Reserved for future assignment by SAE, but available for use by self configurable ECUs

ECU ID (PGN 64965 -ECUID):

Field Name	Value
ECU Part Number	AX140600-03
ECU Serial Number	0000115001
ECU Type	Protocol Converter
ECU Manufacturer Name	Axiomatic

Software ID (PGN 65242 -SOFT):

Field Number	Value
1	LIN - J1939 CAN Converter
2	Project: LIN-J1939CAN-GTH

Figure 20. 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 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

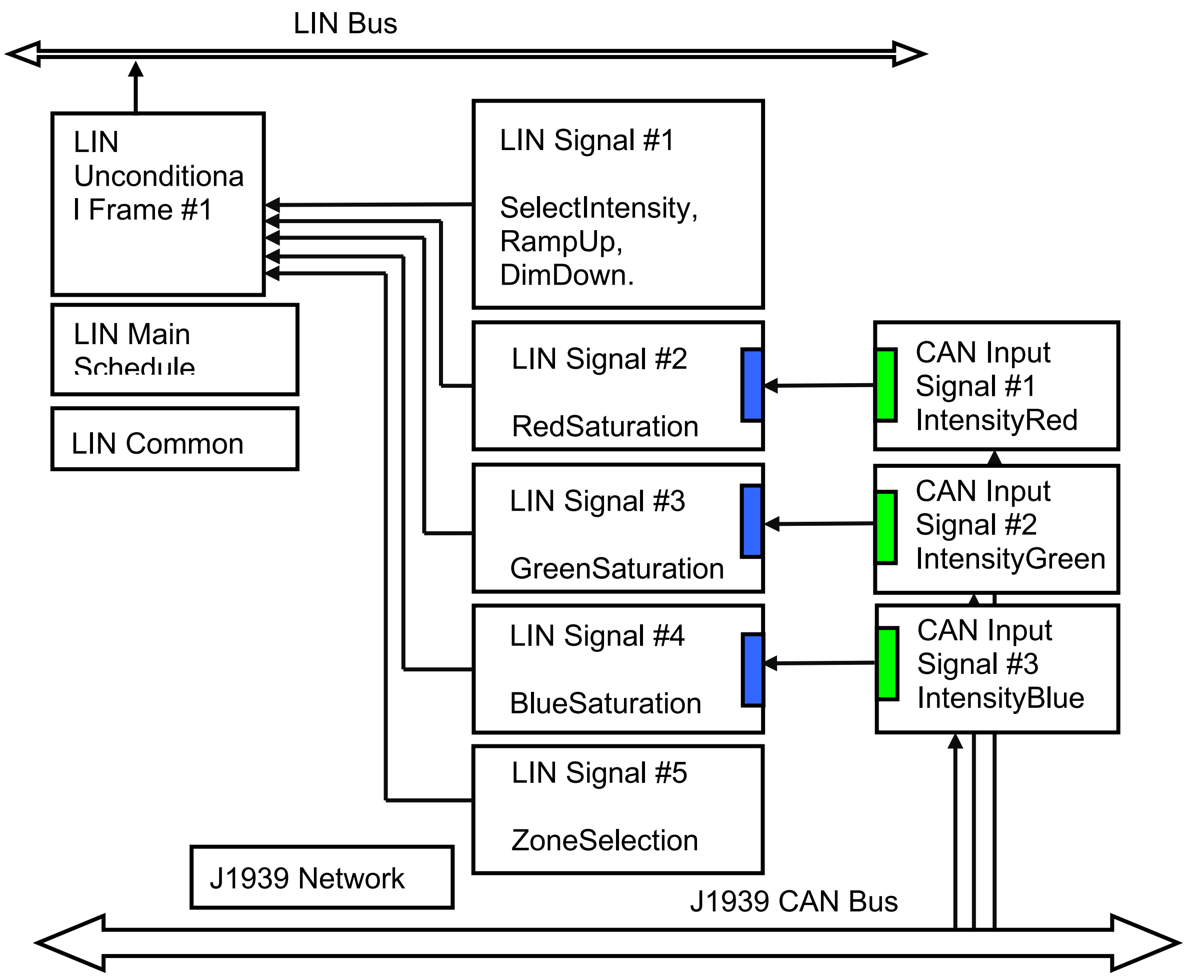
As a first step, create a block diagram of the required converter configuration using the converter function blocks, Figure 21. Then, configure each individual function block, see Figure 21.

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:

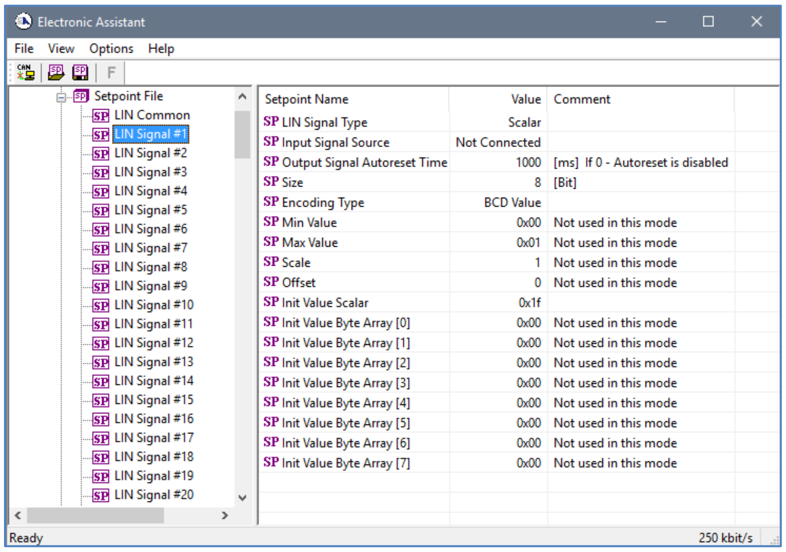
```
MinValue   = 0;
MaxValue   = 0xFF;
Offset     = 0 [%];
Scale      = 100%/0xFF=0.3922 [%/Bit]
```

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.



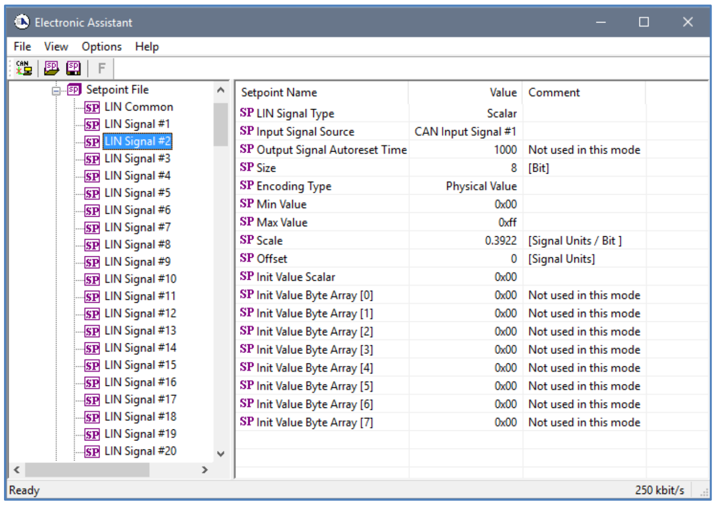
LIN Bus
LIN Unconditional Frame #1
LIN Main Schedule
LIN Common
LIN Signal #1 SelectIntensity, RampUp, DimDown.
LIN Signal #2 RedSaturation
LIN Signal #3 GreenSaturation
LIN Signal #4 BlueSaturation
LIN Signal #5 ZoneSelection
CAN Input Signal #1 IntensityRed
CAN Input Signal #2 IntensityGreen
CAN Input Signal #3 IntensityBlue
J1939 Network
J1939 CAN Bus

Figure 21. Block Diagram of the Example Converter Configuration



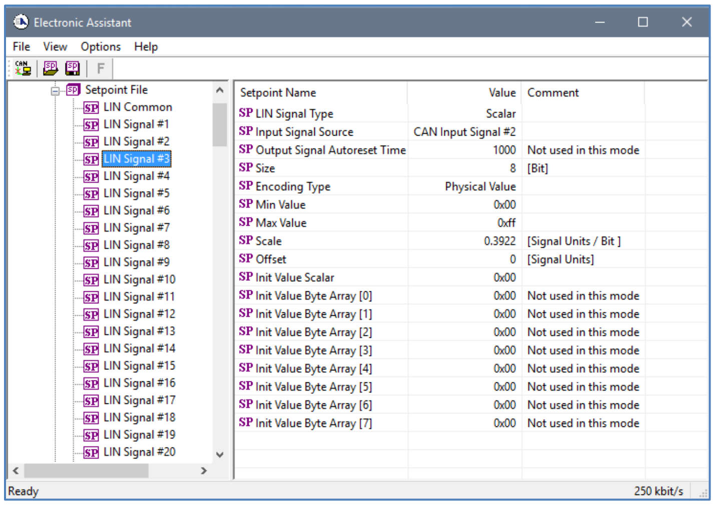
Setpoint Name	Value	Comment
SP LIN Signal Type	Scalar	
SP Input Signal Source	Not Connected	
SP Output Signal Autoreset Time	1000	[ms] If 0 - Autoreset is disabled
SP Size	8	[Bit]
SP Encoding Type	BCD Value	
SP Min Value	0x00	Not used in this mode
SP Max Value	0x01	Not used in this mode
SP Scale	1	Not used in this mode
SP Offset	0	Not used in this mode
SP Init Value Scalar	0x1f	
SP Init Value Byte Array [0]	0x00	Not used in this mode
SP Init Value Byte Array [1]	0x00	Not used in this mode
SP Init Value Byte Array [2]	0x00	Not used in this mode
SP Init Value Byte Array [3]	0x00	Not used in this mode
SP Init Value Byte Array [4]	0x00	Not used in this mode
SP Init Value Byte Array [5]	0x00	Not used in this mode
SP Init Value Byte Array [6]	0x00	Not used in this mode
SP Init Value Byte Array [7]	0x00	Not used in this mode

Figure 22. Example Configuration. LIN Signal #1



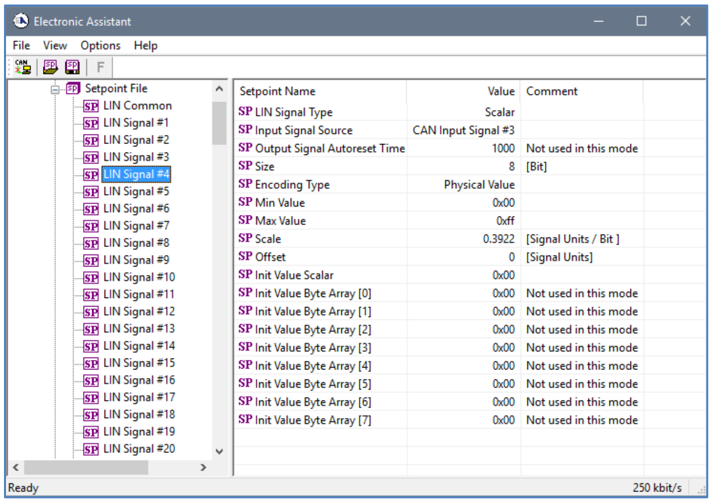
Setpoint Name	Value	Comment
LIN Signal Type	Scalar	
Input Signal Source	CAN Input Signal #1	
Output Signal Autoreset Time	1000	Not used in this mode
Size	8	[Bit]
Encoding Type	Physical Value	
Min Value	0x00	
Max Value	0xff	
Scale	0.3922	[Signal Units / Bit]
Offset	0	[Signal Units]
Init Value Scalar	0x00	
Init Value Byte Array [0]	0x00	Not used in this mode
Init Value Byte Array [1]	0x00	Not used in this mode
Init Value Byte Array [2]	0x00	Not used in this mode
Init Value Byte Array [3]	0x00	Not used in this mode
Init Value Byte Array [4]	0x00	Not used in this mode
Init Value Byte Array [5]	0x00	Not used in this mode
Init Value Byte Array [6]	0x00	Not used in this mode
Init Value Byte Array [7]	0x00	Not used in this mode

Figure 23. Example Configuration. LIN Signal #2



Setpoint Name	Value	Comment
LIN Signal Type	Scalar	
Input Signal Source	CAN Input Signal #2	
Output Signal Autoreset Time	1000	Not used in this mode
Size	8	[Bit]
Encoding Type	Physical Value	
Min Value	0x00	
Max Value	0xff	
Scale	0.3922	[Signal Units / Bit]
Offset	0	[Signal Units]
Init Value Scalar	0x00	
Init Value Byte Array [0]	0x00	Not used in this mode
Init Value Byte Array [1]	0x00	Not used in this mode
Init Value Byte Array [2]	0x00	Not used in this mode
Init Value Byte Array [3]	0x00	Not used in this mode
Init Value Byte Array [4]	0x00	Not used in this mode
Init Value Byte Array [5]	0x00	Not used in this mode
Init Value Byte Array [6]	0x00	Not used in this mode
Init Value Byte Array [7]	0x00	Not used in this mode

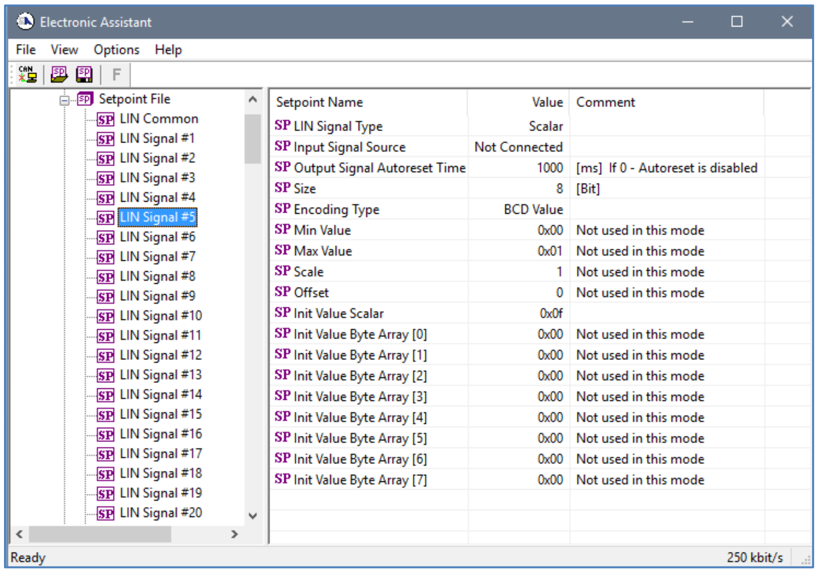
Figure 24. Example Configuration. LIN Signal #3



Setpoint Name	Value	Comment
LIN Signal Type	Scalar	
Input Signal Source	CAN Input Signal #3	
Output Signal Autoreset Time	1000	Not used in this mode
Size	8	[Bit]
Encoding Type	Physical Value	
Min Value	0x00	
Max Value	0xff	
Scale	0.3922	[Signal Units / Bit]
Offset	0	[Signal Units]
Init Value Scalar	0x00	
Init Value Byte Array [0]	0x00	Not used in this mode
Init Value Byte Array [1]	0x00	Not used in this mode
Init Value Byte Array [2]	0x00	Not used in this mode
Init Value Byte Array [3]	0x00	Not used in this mode
Init Value Byte Array [4]	0x00	Not used in this mode
Init Value Byte Array [5]	0x00	Not used in this mode
Init Value Byte Array [6]	0x00	Not used in this mode
Init Value Byte Array [7]	0x00	Not used in this mode

Figure 25. Example Configuration. LIN Signal #4

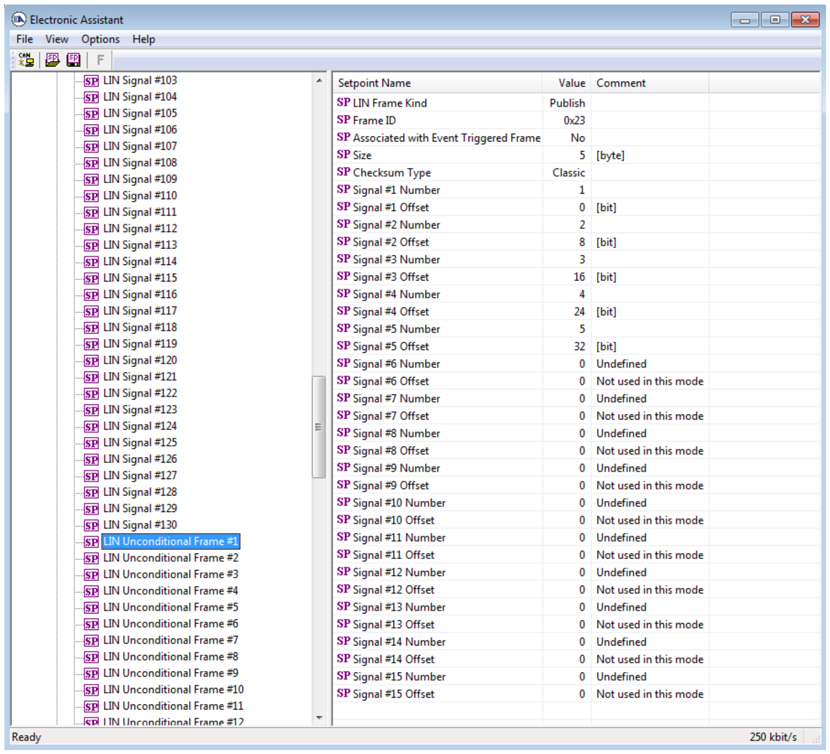
Configure *LIN Signal #5* as ZoneSelection constant signal. Set *Signal Type* to *Scalar*, *Size* to *8 bit*, *Encoding Type* to *BCD Value* and *Init Value Scalar* to *0x0F*.



Setpoint Name	Value	Comment
SP LIN Signal Type	Scalar	
SP Input Signal Source	Not Connected	
SP Output Signal Autoreset Time	1000	[ms] If 0 - Autoreset is disabled
SP Size	8	[Bit]
SP Encoding Type	BCD Value	
SP Min Value	0x00	Not used in this mode
SP Max Value	0x01	Not used in this mode
SP Scale	1	Not used in this mode
SP Offset	0	Not used in this mode
SP Init Value Scalar	0x0f	
SP Init Value Byte Array [0]	0x00	Not used in this mode
SP Init Value Byte Array [1]	0x00	Not used in this mode
SP Init Value Byte Array [2]	0x00	Not used in this mode
SP Init Value Byte Array [3]	0x00	Not used in this mode
SP Init Value Byte Array [4]	0x00	Not used in this mode
SP Init Value Byte Array [5]	0x00	Not used in this mode
SP Init Value Byte Array [6]	0x00	Not used in this mode
SP Init Value Byte Array [7]	0x00	Not used in this mode

Figure 26. Example Configuration. LIN Signal #5

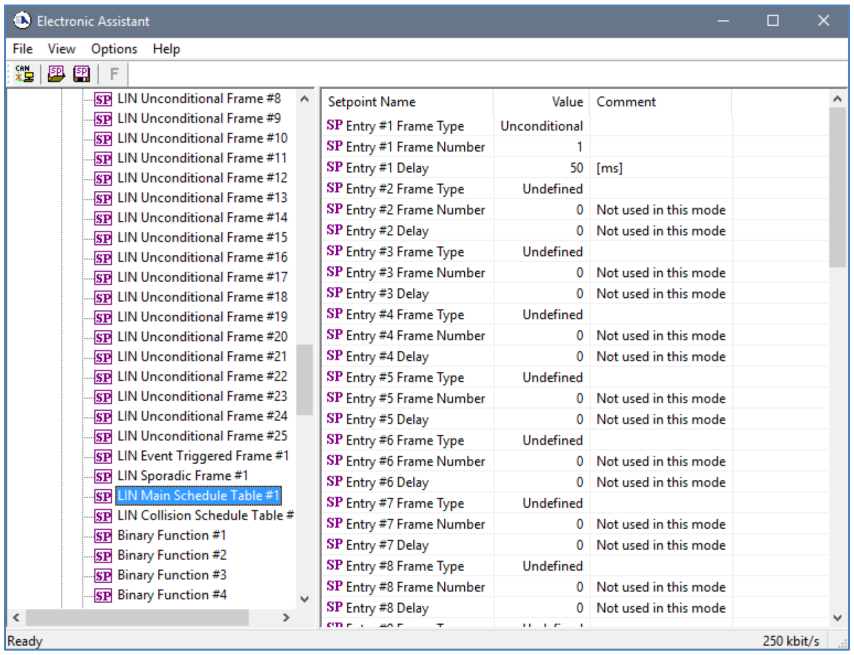
Now, configure *LIN Unconditional Frame #1*. Set *LIN Frame Kind* to *Publish*, *Frame ID* to *0x23*, *Size* to *5 bytes*. Add all previously configured LIN signals to the frame using *(Signal #1…10 Number, Signal #1…10 Offset)* configuration parameter pairs.



Setpoint Name	Value	Comment
SP LIN Frame Kind	Publish	
SP Frame ID	0x23	
SP Associated with Event Triggered Frame	No	
SP Size	5	[byte]
SP Checksum Type	Classic	
SP Signal #1 Number	1	
SP Signal #1 Offset	0	[bit]
SP Signal #2 Number	2	
SP Signal #2 Offset	8	[bit]
SP Signal #3 Number	3	
SP Signal #3 Offset	16	[bit]
SP Signal #4 Number	4	
SP Signal #4 Offset	24	[bit]
SP Signal #5 Number	5	
SP Signal #5 Offset	32	[bit]
SP Signal #6 Number	0	Undefined
SP Signal #6 Offset	0	Not used in this mode
SP Signal #7 Number	0	Undefined
SP Signal #7 Offset	0	Not used in this mode
SP Signal #8 Number	0	Undefined
SP Signal #8 Offset	0	Not used in this mode
SP Signal #9 Number	0	Undefined
SP Signal #9 Offset	0	Not used in this mode
SP Signal #10 Number	0	Undefined
SP Signal #10 Offset	0	Not used in this mode
SP Signal #11 Number	0	Undefined
SP Signal #11 Offset	0	Not used in this mode
SP Signal #12 Number	0	Undefined
SP Signal #12 Offset	0	Not used in this mode
SP Signal #13 Number	0	Undefined
SP Signal #13 Offset	0	Not used in this mode
SP Signal #14 Number	0	Undefined
SP Signal #14 Offset	0	Not used in this mode
SP Signal #15 Number	0	Undefined
SP Signal #15 Offset	0	Not used in this mode

Figure 27. Example Configuration. LIN Unconditional Frame #1

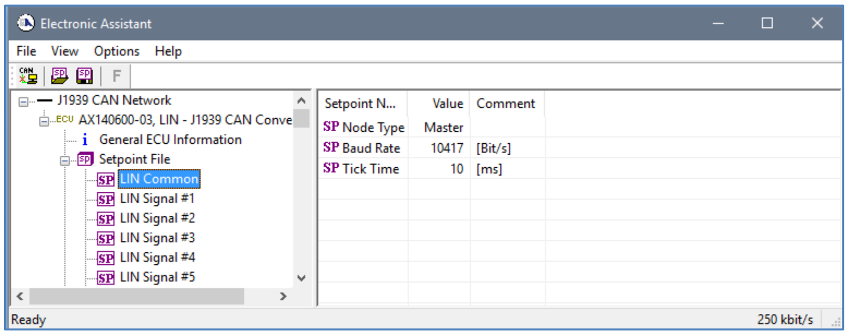
Set *LIN Schedule Table #1* with only one entry: *LIN Unconditional Frame #1*. Set *Delay* to *50 ms*, see Figure 28.


Electronic Assistant
File View Options Help
LIN Unconditional Frame #8
LIN Unconditional Frame #9
LIN Unconditional Frame #10
LIN Unconditional Frame #11
LIN Unconditional Frame #12
LIN Unconditional Frame #13
LIN Unconditional Frame #14
LIN Unconditional Frame #15
LIN Unconditional Frame #16
LIN Unconditional Frame #17
LIN Unconditional Frame #18
LIN Unconditional Frame #19
LIN Unconditional Frame #20
LIN Unconditional Frame #21
LIN Unconditional Frame #22
LIN Unconditional Frame #23
LIN Unconditional Frame #24
LIN Unconditional Frame #25
LIN Event Triggered Frame #1
LIN Sporadic Frame #1
LIN Main Schedule Table #1
LIN Collision Schedule Table #
Binary Function #1
Binary Function #2
Binary Function #3
Binary Function #4
Ready
250 kbit/s

Setpoint Name	Value	Comment
SP Entry #1 Frame Type	Unconditional	
SP Entry #1 Frame Number	1	
SP Entry #1 Delay	50	[ms]
SP Entry #2 Frame Type	Undefined	
SP Entry #2 Frame Number	0	Not used in this mode
SP Entry #2 Delay	0	Not used in this mode
SP Entry #3 Frame Type	Undefined	
SP Entry #3 Frame Number	0	Not used in this mode
SP Entry #3 Delay	0	Not used in this mode
SP Entry #4 Frame Type	Undefined	
SP Entry #4 Frame Number	0	Not used in this mode
SP Entry #4 Delay	0	Not used in this mode
SP Entry #5 Frame Type	Undefined	
SP Entry #5 Frame Number	0	Not used in this mode
SP Entry #5 Delay	0	Not used in this mode
SP Entry #6 Frame Type	Undefined	
SP Entry #6 Frame Number	0	Not used in this mode
SP Entry #6 Delay	0	Not used in this mode
SP Entry #7 Frame Type	Undefined	
SP Entry #7 Frame Number	0	Not used in this mode
SP Entry #7 Delay	0	Not used in this mode
SP Entry #8 Frame Type	Undefined	
SP Entry #8 Frame Number	0	Not used in this mode
SP Entry #8 Delay	0	Not used in this mode

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*.

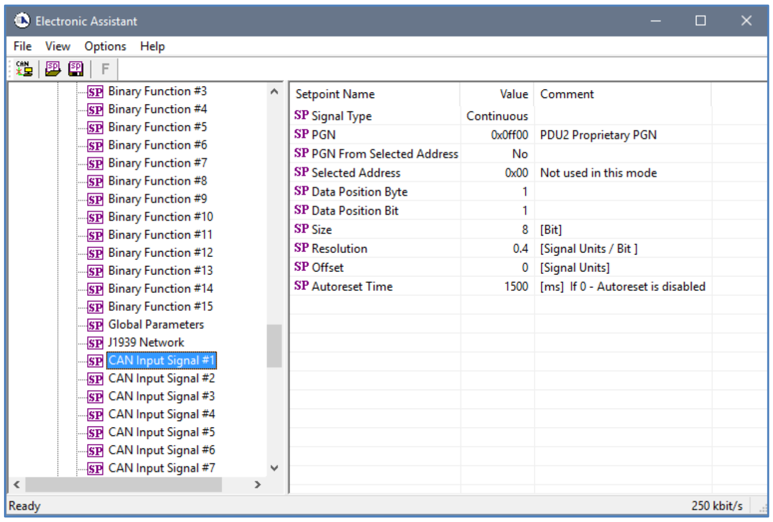


Setpoint N...	Value	Comment
SP Node Type	Master	
SP Baud Rate	10417	[Bit/s]
SP Tick Time	10	[ms]

Figure 29. Example Configuration. LIN Common

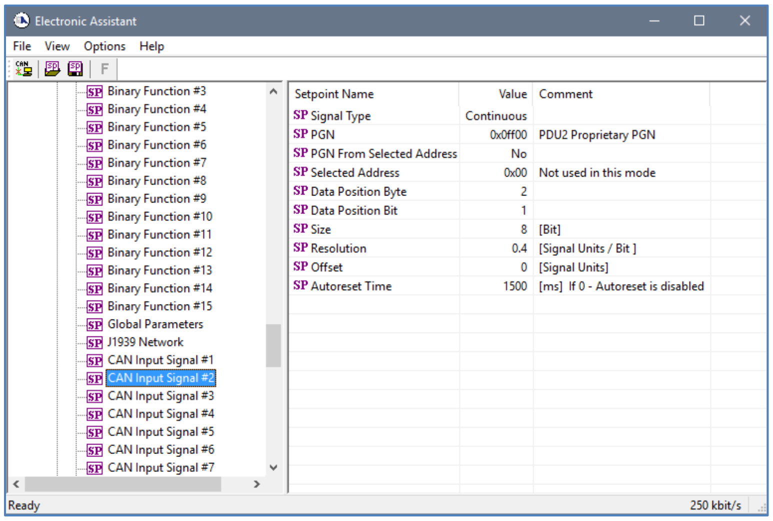
The LIN bus is now running. Configure CAN bus input signals. All three input signals will parse one CAN message. Start from *CAN Input Signal #1* and configure it as IntensityRed signal. The *Autoreset Time* set to *1500 ms* (1.5 seconds) to ensure that the LED will not switch off in case one or two 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.



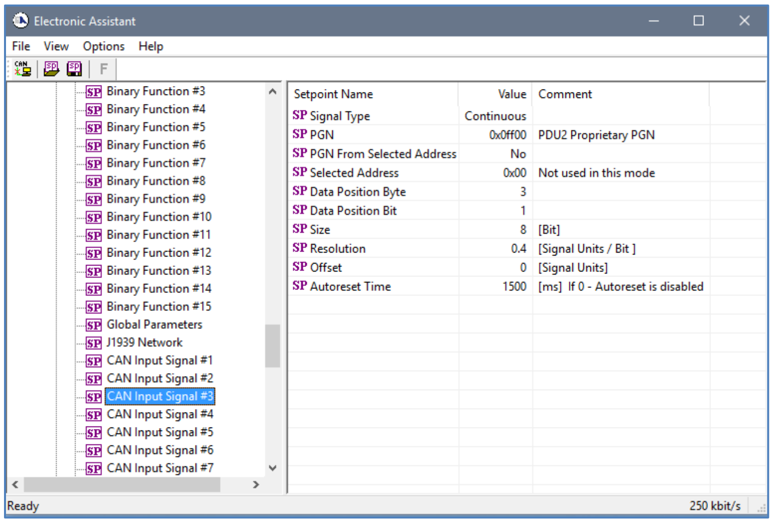
Setpoint Name	Value	Comment
SP Signal Type	Continuous	
SP PGN	0x0ff00	PDU2 Proprietary PGN
SP PGN From Selected Address	No	
SP Selected Address	0x00	Not used in this mode
SP Data Position Byte	1	
SP Data Position Bit	1	
SP Size	8	[Bit]
SP Resolution	0.4	[Signal Units / Bit]
SP Offset	0	[Signal Units]
SP Autoreset Time	1500	[ms] If 0 - Autoreset is disabled

Figure 30. Example Configuration. CAN Input Signal #1



Setpoint Name	Value	Comment
SP Signal Type	Continuous	
SP PGN	0x0ff00	PDU2 Proprietary PGN
SP PGN From Selected Address	No	
SP Selected Address	0x00	Not used in this mode
SP Data Position Byte	2	
SP Data Position Bit	1	
SP Size	8	[Bit]
SP Resolution	0.4	[Signal Units / Bit]
SP Offset	0	[Signal Units]
SP Autoreset Time	1500	[ms] If 0 - Autoreset is disabled

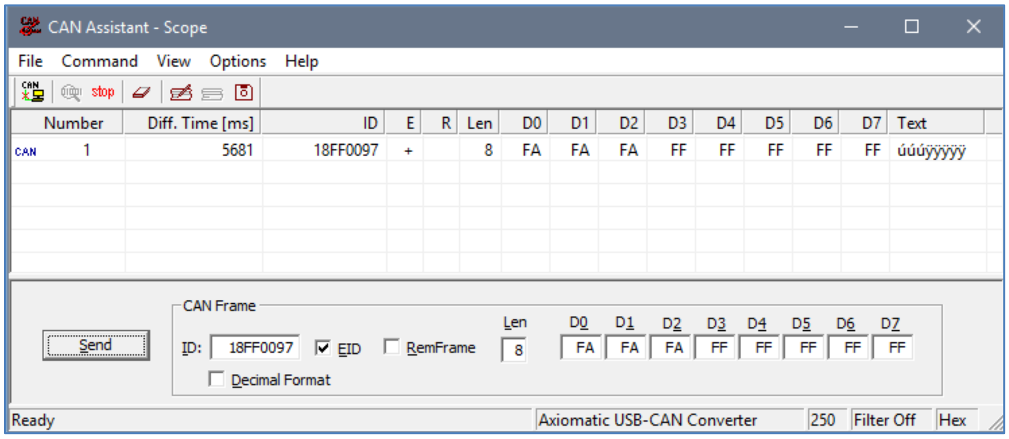
Figure 31. Example Configuration. CAN Input Signal #2



Setpoint Name	Value	Comment
SP Signal Type	Continuous	
SP PGN	0x0ff00	PDU2 Proprietary PGN
SP PGN From Selected Address	No	
SP Selected Address	0x00	Not used in this mode
SP Data Position Byte	3	
SP Data Position Bit	1	
SP Size	8	[Bit]
SP Resolution	0.4	[Signal Units / Bit]
SP Offset	0	[Signal Units]
SP Autoreset Time	1500	[ms] If 0 - Autoreset is disabled

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.



Number	Diff. Time [ms]	ID	E	R	Len	D0	D1	D2	D3	D4	D5	D6	D7	Text
CAN 1	5681	18FF0097	+		8	FA	FA	FA	FF	FF	FF	FF	FF	úúúÿÿÿÿÿ

CAN Frame

ID: 18FF0097 ☑ EID ☐ RemFrame ☐ Decimal Format

Len	D0	D1	D2	D3	D4	D5	D6	D7
8	FA	FA	FA	FF	FF	FF	FF	FF

Figure 33. Example Configuration. Generating CAN Test Message

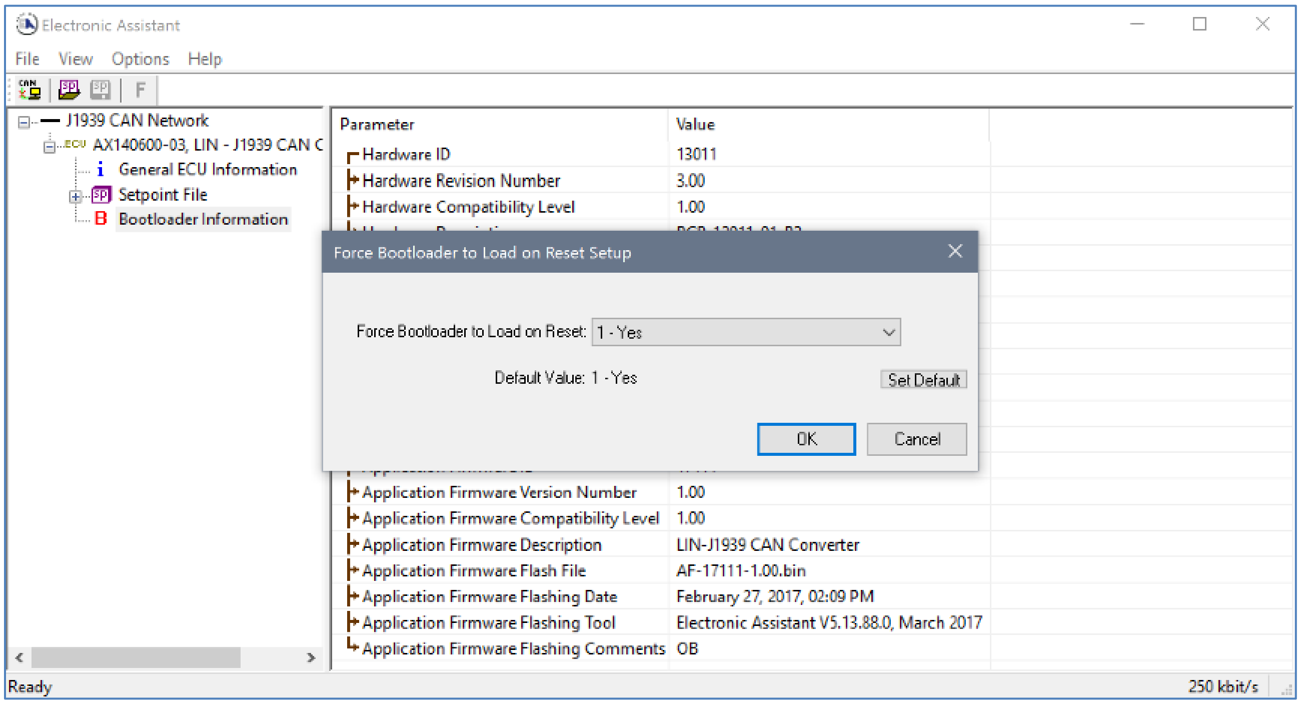
The LED will stay on for 1.5 seconds and then switch off, if the message is not being retransmitted, as per the user requirements.

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

5 FLASHING NEW FIRMWARE

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 34.[1]



Parameter	Value
Hardware ID	13011
Hardware Revision Number	3.00
Hardware Compatibility Level	1.00
Application Firmware Version Number	1.00
Application Firmware Compatibility Level	1.00
Application Firmware Description	LIN-J1939 CAN Converter
Application Firmware Flash File	AF-17111-1.00.bin
Application Firmware Flashing Date	February 27, 2017, 02:09 PM
Application Firmware Flashing Tool	Electronic Assistant V5.13.88.0, March 2017
Application Firmware Flashing Comments	OB

Force Bootloader to Load on Reset Setup

Force Bootloader to Load on Reset: 1 - Yes

Default Value: 1 - Yes

Set Default

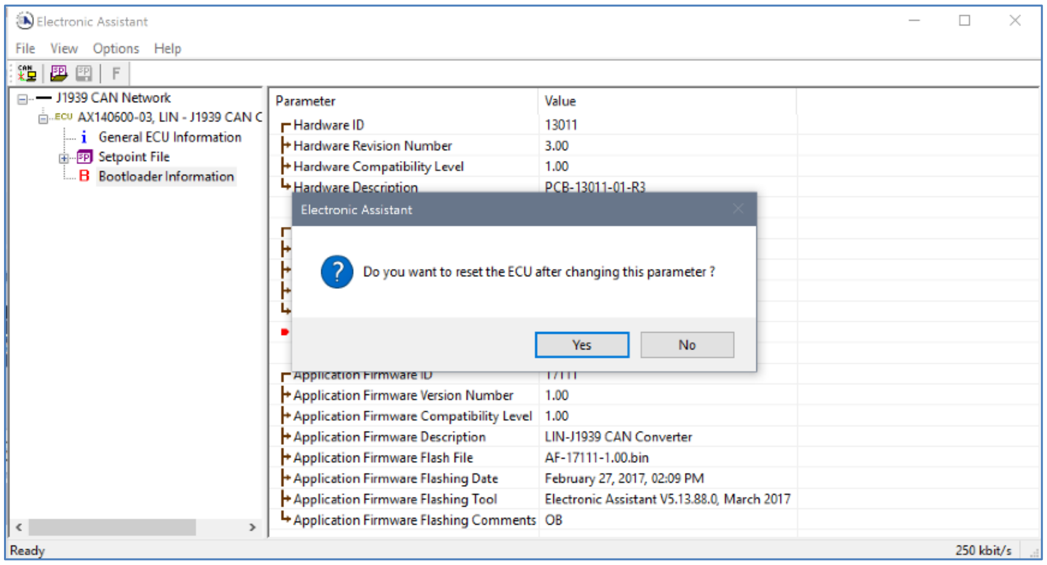
OK Cancel

Figure 34. Bootloader Activation. First Step

[1] *Bootloaders V1.xx,…,2.xx originally shipped with application firmware V1.xx (check the Bootloader Information group screen) do not have the automatic baud rate detection mechanism. A special application firmware compiled specifically for these bootloaders should be used to upgrade the converter firmware. The CAN baud rate should be set to 250 kbit/s (or 500 kbit/s for 500 kbit/s bootloader version) in the EA Options menu, otherwise the Force Bootloader to Load on Reset parameter will not be activated.*

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 35. Select *Yes*.

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.



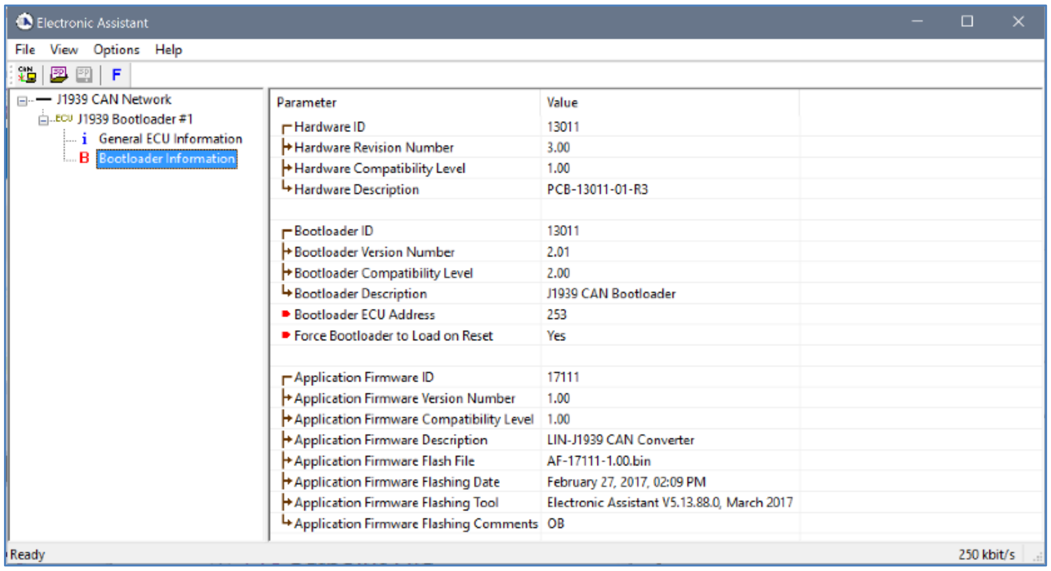
Parameter	Value
Hardware ID	13011
Hardware Revision Number	3.00
Hardware Compatibility Level	1.00
Hardware Description	PCB-13011-01-R3
Application Firmware ID	17111
Application Firmware Version Number	1.00
Application Firmware Compatibility Level	1.00
Application Firmware Description	LIN-J1939 CAN Converter
Application Firmware Flash File	AF-17111-1.00.bin
Application Firmware Flashing Date	February 27, 2017, 02:09 PM
Application Firmware Flashing Tool	Electronic Assistant V5.13.88.0, March 2017
Application Firmware Flashing Comments	OB

Electronic Assistant

Do you want to reset the ECU after changing this parameter ?

Figure 35. 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 36. The information can be slightly different for different versions of the bootloader.

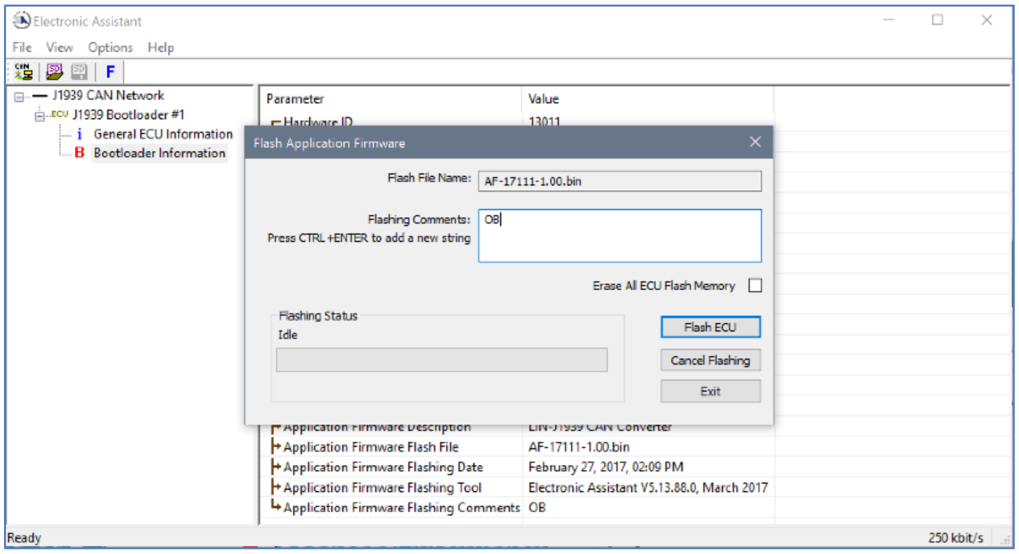


Parameter	Value
Hardware ID	13011
Hardware Revision Number	3.00
Hardware Compatibility Level	1.00
Hardware Description	PCB-13011-01-R3
Bootloader ID	13011
Bootloader Version Number	2.01
Bootloader Compatibility Level	2.00
Bootloader Description	J1939 CAN Bootloader
Bootloader ECU Address	253
Force Bootloader to Load on Reset	Yes
Application Firmware ID	17111
Application Firmware Version Number	1.00
Application Firmware Compatibility Level	1.00
Application Firmware Description	LIN-J1939 CAN Converter
Application Firmware Flash File	AF-17111-1.00.bin
Application Firmware Flashing Date	February 27, 2017, 02:09 PM
Application Firmware Flashing Tool	Electronic Assistant V5.13.88.0, March 2017
Application Firmware Flashing Comments	OB

Figure 36. 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 37.


Electronic Assistant

File View Options Help

J1939 CAN Network
J1939 Bootloader #1
General ECU Information
Bootloader Information

Parameter	Value
Hardware ID	13011
Application Firmware Description	LIN-J1939 CAN Converter
Application Firmware Flash File	AF-17111-1.00.bin
Application Firmware Flashing Date	February 27, 2017, 02:09 PM
Application Firmware Flashing Tool	Electronic Assistant V5.13.88.0, March 2017
Application Firmware Flashing Comments	OB

Flash Application Firmware

Flash File Name: AF-17111-1.00.bin

Flashing Comments: OB
Press CTRL+ENTER to add a new string

Erase All ECU Flash Memory ☐

Flashing Status
Idle

Flash ECU
Cancel Flashing
Exit

Ready 250 kbit/s

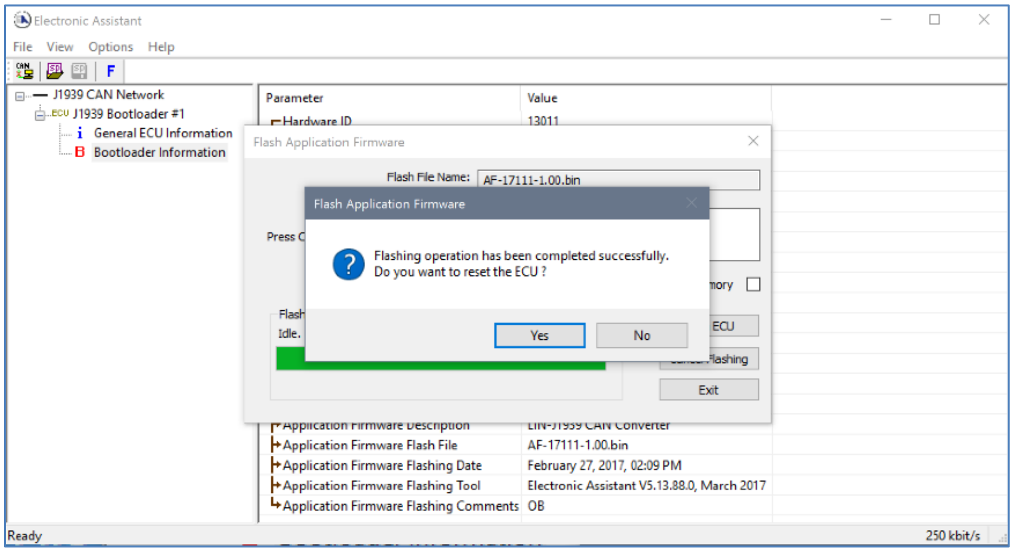
Figure 37. Flashing New Firmware. Preparation

[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 set by the old firmware and force the converter to load the default values after flashing the new firmware. Otherwise, the default 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 38.


Electronic Assistant

File View Options Help

J1939 CAN Network
J1939 Bootloader #1
General ECU Information
Bootloader Information

Parameter	Value
Hardware ID	13011
Application Firmware Description	LIN-J1939 CAN Converter
Application Firmware Flash File	AF-17111-1.00.bin
Application Firmware Flashing Date	February 27, 2017, 02:09 PM
Application Firmware Flashing Tool	Electronic Assistant V5.13.88.0, March 2017
Application Firmware Flashing Comments	OB

Flash Application Firmware

Flash File Name: AF-17111-1.00.bin

Flash Application Firmware

Flashing operation has been completed successfully.
Do you want to reset the ECU ?

Yes No

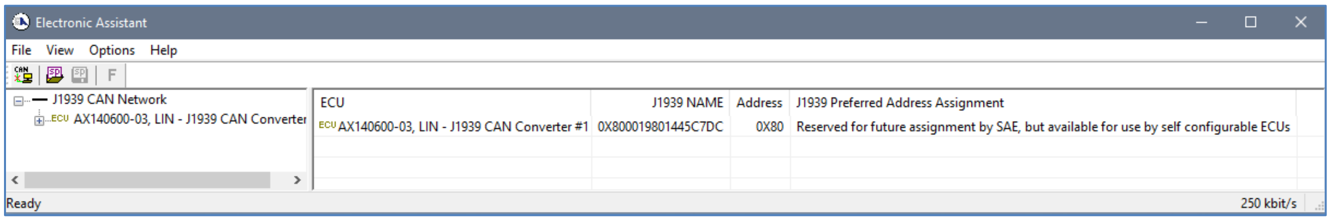
Idle.

Exit

Ready 250 kbit/s

Figure 38. Flashing New Firmware. Final Reset.

Select *Yes* and see the ECU running the new firmware, see Figure 39. This will indicate that the flashing operation has been performed successfully.



ECU	J1939 NAME	Address	J1939 Preferred Address Assignment
AX140600-03, LIN - J1939 CAN Converter #1	0X800019801445C7DC	0X80	Reserved for future assignment by SAE, but available for use by self configurable ECUs

Figure 39. 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 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, 2.4…20 kbit/s)
CAN Port	1 CAN (SAE J1939, 250kbit/s, 500kbit/s, 667kbit/s, 1Mbit/s. Automatic Baud Rate Detection)
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 TE Deutsch type connector Refer to dimensional drawing (below).
Electrical Connections	Integral 6-pin connector (equivalent TE Deutsch P/N: DT04-6P) A mating plug kit is available as Axiomatic P/N: AX070119

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	Via the Axiomatic Electronic Assistant, P/Ns: AX070502 or AX070506K
User Interface	Parameters are configurable using the Axiomatic Electronic Assistant.

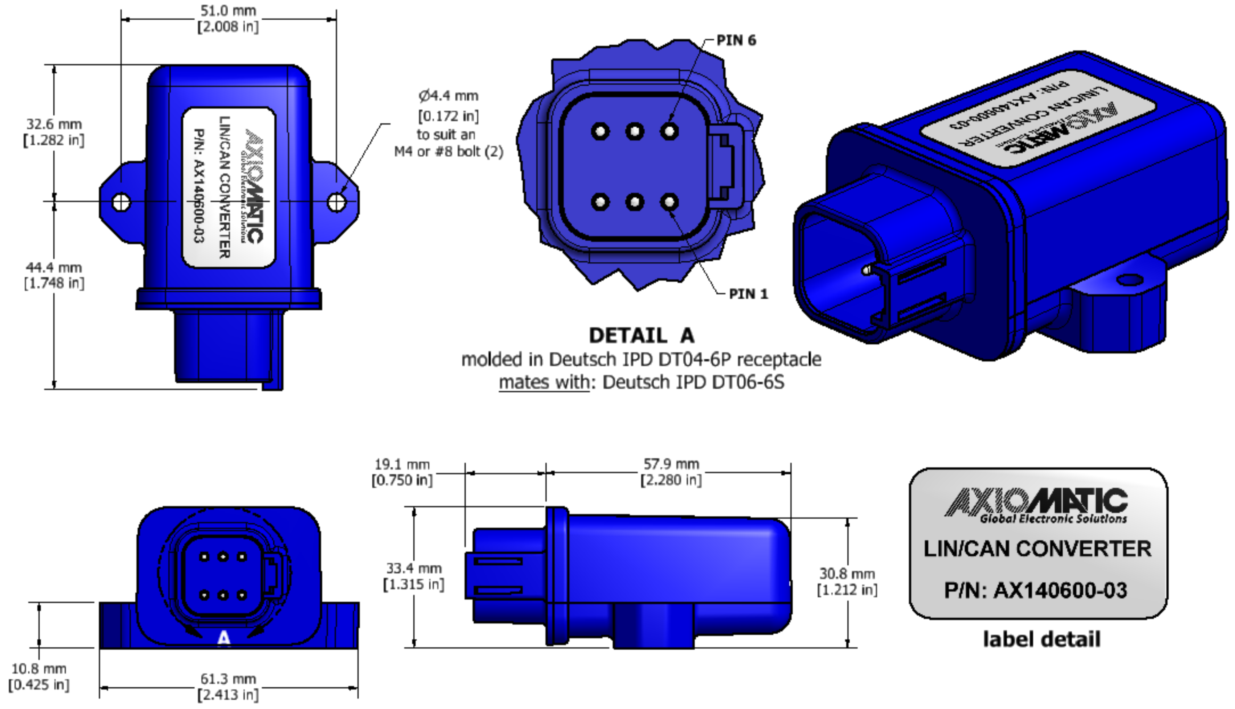


Figure 40. Unit Dimensions

7 VERSION HISTORY

User Manual Version	Firmware version	Axiomatic Electronic Assistant (EA) version	Date	Author	Modifications
2C	2.xx	5.15.108.0 or higher	Sept. 12, 2023	Kiril Mojsov	• Performed Legacy Updates
2B	2.xx	5.15.108.0 or higher	Sept. 14, 2020	Olek Bogush	• Added notes excluding Proprietary A PGN (61184) from customer's use in function blocks. Updated *CAN Standard Implementation* table, *CAN Input Signal* and *CAN Output Message* function blocks.
2A	2.xx	5.15.108.0 or higher	June 3, 2020	Olek Bogush	• Updated Finnish office phone number on the front page.
2	2.xx	5.15.108.0 or higher	December 11, 2019	Olek Bogush	• Added automatic baud rate detection from the list of 250kbit/s, 500kbit/s, 667kbit/s, 1Mbit/s baud rates. Updated relevant sections of the UM. • Updated *J1939 Network* sub-section. Added *Baud Rate and Automatic Baud Rate Detection* configuration parameters. Updated *Slew Rate* configuration parameter description. • Increased number of LIN signals from 75 to 130. Updated *LIN Signal* function block. • Increased number of LIN signals associated with each LIN frame from 10 to 15. Updated *LIN Unconditional Frame* function block. • Added *Use CAN Signals in Another CAN Message* to *CAN Output Message* function block to share CAN signals. • Added *Transmit on LIN Unconditional Frame Number* configuration parameters to *CAN Output Message* function block to transmit CAN messages on successful transmission or reception of LIN frames.
1B	1.xx	5.13.88.1 or higher	April 7, 2017	Olek Bogush	• Corrected EA version number.
1A	1.xx	5.13.88.1 or higher	April 3, 2017	Olek Bogush	• Corrected *LIN Standard Implementation* table.
1	1.xx	5.13.88.1 or higher	February 27, 2017	Olek Bogush	• Initial release. Based on UMAX140600 Version 2B.

AXIOMATIC
Global Electronic Solutions

OUR PRODUCTS

AC/DC Power Supplies

Actuator Controls/Interfaces

Automotive Ethernet Interfaces

Battery Chargers

CAN Controls, Routers, Repeaters

CAN/WiFi, CAN/Bluetooth, Routers

Current/Voltage/PWM Converters

DC/DC Power Converters

Engine Temperature Scanners

Ethernet/CAN Converters, Gateways, Switches

Fan Drive Controllers

Gateways, CAN/Modbus, RS-232

Gyroscopes, Inclinometers

Hydraulic Valve Controllers

Inclinometers, Triaxial

I/O Controls

LVDT Signal Converters

Machine Controls

Modbus, RS-422, RS-485 Controls

Motor Controls, Inverters

Power Supplies, DC/DC, AC/DC

PWM Signal Converters/Isolators

Resolver Signal Conditioners

Service Tools

Signal Conditioners, Converters

Strain Gauge CAN Controls

Surge Suppressors

OUR COMPANY

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

QUALITY DESIGN AND MANUFACTURING

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

WARRANTY, APPLICATION APPROVALS/LIMITATIONS

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

COMPLIANCE

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

SAFE USE

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

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

SERVICE

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

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

DISPOSAL

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

CONTACTS

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

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