

## Application-Oriented Manual

SAE J1939 STX API

60880884

We automate your success.

Item number: 60880884

Revision 1.03

2021/11/09 / Printed in Germany

This document has been compiled by Jetter AG with due diligence, and based on the known state of the art.

In the case of modifications, further developments or enhancements to products shipped in the past, a revised document will be supplied only if required by law, or deemed appropriate by Jetter AG. Jetter AG shall neither be liable nor responsible for any errors in form or content, lacks in updating and possibly resulting damages or disadvantages.

The logos, brand names, and product names mentioned in this document are trademarks of Jetter AG, of associated companies or other title owners and must not be used without consent of the respective title owner.

# Table of contents

<b>1</b>	<b>SAE J1939</b>	<b>5</b>
	J1939 protocol - Layer description.....	6
<b>2</b>	<b>SAE J1939-STX-API</b>	<b>9</b>
	STX function SAEJ1939Init().....	10
	STX function SAEJ1939Exit().....	11
	STX function SAEJ1939SetSA().....	13
	STX function SAEJ1939GetSA().....	14
	STX function SAEJ1939AddRx().....	15
	STX function SAEJ1939AddTx().....	18
	STX function SAEJ1939RequestPGN().....	21
	STX function SAEJ1939GetDM1().....	24
	STX function SAEJ1939GetDM2().....	27
	STX function SAEJ1939SetSPNConversion().....	30
	STX function SAEJ1939GetSPNConversion().....	31



---

# 1 SAE J1939

---

**Introduction**

This chapter describes the J1939 layer and the structure of a J1939 message.

---

**Content**

<b>Topic</b>	<b>Page</b>
J1939 protocol - Layer description .....	6

## J1939 protocol - Layer description

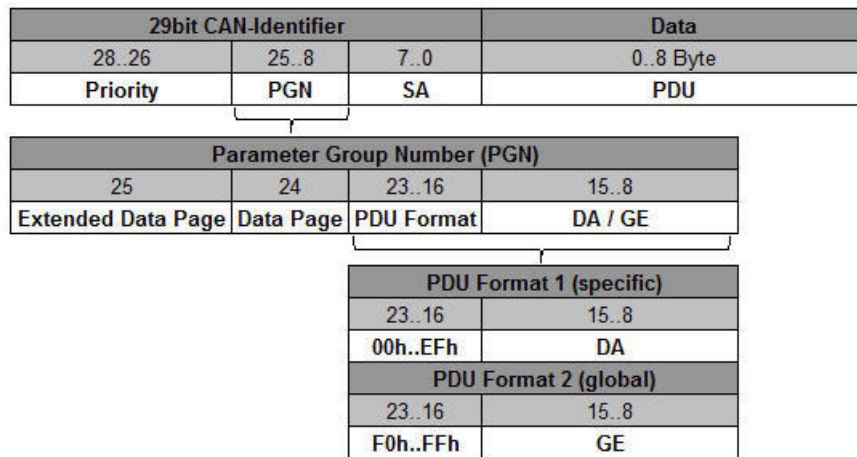
### Physical layer

The protocol SAE J1939 is based on the CAN bus and uses as physical layer CAN Highspeed to ISO 11898.

- Baud rate 250 kBit
- 30 nodes max.
- 2-wire line with a terminating resistor of 120 Ω
- Bus length (without tap line) 40 m
- Max. tap line length 1 m

### Content of a J1939 Message

The following diagram shows the structure of a J1939 message:



Abbreviation	Description
DA	Destination Address
GE	Group Extensions
PDU	Protocol Data Unit
PGN	Parameter Group Number
SA	Source Address

### Identifier structure

The following example shows the structure of an identifier (hexadecimal):  
0x18FEE927

Identifier component	Description
27	Source Address
FEE9	Parameter Group Number
18	Priority

**Meaning of SPN - Suspect Parameter Number**

The SPN is a number defined by the SAE J1939 standard containing individual parameters (e.g. engine RPM) as standardized message.

Below is an example of SPN parameters:

**spn110 - Engine Coolant Temperature - Temperature of the engine coolant.**

Data length:	1 bytes
Resolution:	1 °C/bit , -40 °C offset
Data range:	-40 ... 210 °C
Type:	Measured value
Suspect Parameter Number:	110
Vehicle Application Layer - J1939-71 (J1939-71 Rev. Aug 2002)	
Parameter Group Number:	[65262]

**Meaning of the Parameter Group Number (PGN)**

The PGN is a number defined in the SAE J1939 standard that groups together several SPNs into a meaningful group. The PGN is part of the CAN identifier. The 8-byte data (PDU) contain the values of individual SPNs.

The example below shows a PGN 65262 (0xFEEE):

**PGN 65262 Engine Temperature 1 - ET1**

Part of the PGN	Value	Comment
Transmission Repetition Rate	1 s	
Data Length	8	
Extended Data Page	0	
Data Page	0	
PDU Format	254	
PDU Specific	238	PGN Supporting Information
Default Priority	6	
Parameter Group Number	65262	in hex: 0xFEEE

Start position	Length	Parameter name	SPN
1	1 bytes	Engine Coolant Temperature	110
2	1 bytes	Engine Fuel Temperature 1	174
3 - 4	2 bytes	Engine Oil Temperature 1	175
5 - 6	2 bytes	Engine Turbocharger Oil Temperature	176
7	1 bytes	Engine Intercooler Temperature	52
8	1 bytes	Engine Intercooler Thermostat Opening	1134





## 2 SAE J1939-STX-API

<b>Introduction</b>	This chapter describes the STX functions of the SAE J1939 STX API.
<b>The SAE J1939 Standard</b>	SAE J1939 is an open standard for networking and communication in the commercial vehicle sector. The focal point of the application is the networking of the power train and chassis. The J1939 protocol originates from the international Society of Automotive Engineers (SAE) and works on the physical layer with CAN high-speed according to ISO 11898.
<b>Application</b>	STX functions are used in the communication between an SAE J1939-compliant controller and other ECUs in the vehicle. As a rule, engine data, such as RPM, speed or coolant temperature are read out and displayed.
<b>Documentation</b>	The key SAE J1939 specifications are: <ul style="list-style-type: none"> <li>▪ J1939-11 - Information on the physical layer</li> <li>▪ J1939-21 - Information on the data link layer</li> <li>▪ J1939-71 - Information on the application layer vehicles</li> <li>▪ J1939-73 - Information on the application layer range analysis</li> <li>▪ J1939-81 - Network management</li> </ul>

**Devices** The following devices support the SAE J1939-STX API functions:

Type of device	Designation
Controller	JCM-350-E01/E02, JCM-350-E03, JCM-620 JCM-52x, JCM-511, LMC-x, JCM-63x
HMI	BTM07, BTM09(B), BTM010, BTM011(B), BTM012 JVM-104, JVM-407(B), JVM-507(B), JVM-604(B)

### Content

Topic	Page
STX function SAEJ1939Init().....	10
STX function SAEJ1939Exit().....	11
STX function SAEJ1939SetSA() .....	13
STX function SAEJ1939GetSA() .....	14
STX function SAEJ1939AddRx().....	15
STX function SAEJ1939AddTx() .....	18
STX function SAEJ1939RequestPGN() .....	21
STX function SAEJ1939GetDM1() .....	24
STX function SAEJ1939GetDM2() .....	27
STX function SAEJ1939SetSPNConversion() .....	30
STX function SAEJ1939GetSPNConversion() .....	31

---

## STX function SAEJ1939Init()

---

**Introduction**

Calling the `SAEJ1939Init()` function initializes one of the existing CAN busses for the J1939 protocol (not CAN 0 if no Ethernet - interface is available, since CAN 0 is reserved for CANopen<sup>®</sup>). From then on, the device has the SA (Source Address) assigned by the function parameter `mySA`. Thus, it has got its own device address on the bus.

**Function declaration**

```
Function SAEJ1939Init(
    CANNNo: Int,
    mySA: Byte,
) : Int;
```

**Function parameters**

The `SAEJ1939Init()` function has the following parameters:

Parameter	Description	Value
CANNNo	CAN channel number	0/1 ... CANMAX
mySA	Own source address	0 ... 253

**Return value**

This function transmits the following return values to the higher-level program.

**Return value**

0	OK
-1	Error when checking parameters
-3	Insufficient memory for SAE J1939

**CANNNo parameter**

This parameter specifies the number of the SAEJ1939 interface. `CANNNo = 0/1` is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

**How to apply this function**

Initializing the CAN bus 1. The device has SA 20 (0x14). The device can now send messages with the set SA (and only these messages).

```
Result := SAEJ1939Init(1, 20);
```

**Address Claiming**

*Address Claiming* has not been implemented.

---

## STX function SAEJ1939Exit()

### Introduction

The initialized CAN interface is closed by calling the `SAEJ1939Exit()` function.

### Function declaration

```
Function SAEJ1939Exit (
    CANNo: Int,
) : Int;
```

### Function parameters

The `SAEJ1939Exit()` function has the following parameters:

Parameter	Description	Value
CANNo	Number of the CAN interface	0 ... CANMAX

### Return value

This function passes the following return values to the higher-level program:

#### Return value

0	OK
-1	Error when checking parameters

### CANNo parameter

This parameter specifies the number of the CAN interface. `CANNo = 0` is assigned to the first interface. The number of CAN interfaces depends on the device. For information on the maximum number of CAN interfaces (`CANMAX`) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

## 2 SAE J1939-STX-API

---

### How to apply this function

Closing the CAN interface 0.

```
Result := SAEJ1939Exit(0);
```

---

### Operating principle

The function closes the CAN interface for the passed interface number. PGNs are no longer sent/received. If the interface is to be used again afterwards, SAEJ1939Init() must be called. Subsequently, all PGNs must also be registered again (SAEJ1939AddRx and SAEJ1939AddTx).

---

### Related topics

- STX function SAEJ1939Init()
-

---

## STX function SAEJ1939SetSA()

---

**Introduction**

The SAEJ1939SetSA() function lets you change your own SA (Source Address) during runtime.

**Function declaration**

```
Function SAEJ1939SetSA (
    CANNNo: Int,
    mySA: Byte,
) : Int;
```

**Function parameters**

The SAEJ1939SetSA () function has the following parameters:

Parameter	Description	Value
CANNNo	CAN channel number	0/1 ... CANMAX
mySA	New SA	0 ... 253

**Return value**

This function transmits the following return values to the higher-level program.

**Return value**

0	OK
-1	Error when checking parameters

**CANNNo parameter**

This parameter specifies the number of the SAEJ1939 interface. CANNNo = 0/1 is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

**How to apply this function**

Changing the SA during runtime.

```
Result := SAEJ1939SetSA(1, 20);
```

**Important note**

Messages are immediately sent/received using the new SA.

---

---

## STX function SAEJ1939GetSA()

---

**Introduction** The SAEJ1939GetSA() function lets you determine your own SA (Source Address).

**Function declaration**

```
Function SAEJ1939GetSA(
    CANNo: Int,
    ref mySA: Byte,
) : Int;
```

**Function parameters** The SAEJ1939GetSA () function has the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	0/1 ... CANMAX
mySA	SA currently set	0 ... 253

**Return value** This function transmits the following return values to the higher-level program.

---

**Return value**

0	OK
-1	Error when checking parameters

**CANNo parameter**

This parameter specifies the number of the SAEJ1939 interface. CANNo = 0/1 is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

**How to apply this function**

This function returns the currently set SA.

```
Result := SAEJ1939SetSA(1, actual_SA);
```

---

## STX function SAEJ1939AddRx()

### Introduction

Calling the `SAEJ1939AddRx()` function requests the device to receive a specific message. This message is sent from another bus node. The address of this bus node is transferred to this function as a `bySA` parameter. If the message is not sent, the value received last remains valid. Cyclic reading continues until the `SAEJ1939Init()` function is called again.

### Function declaration

```
Function SAEJ1939AddRx (
    CANNo: Int,
    IPGN: Long,
    bySA: Byte,
    BytePos: Int,
    BitPos: Int,
    DataType: Int,
    DataLength: Int,
    const ref VarAddr,
    ref stJ1939: J1939Rx
    EventTime: Int,
    InhibitTime: Int,
) : Int;
```

### Function parameters

The `SAEJ1939AddRx()` function has the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	0/1 ... CANMAX
IPGN	PGN Parameter Group Number	0 ... 0x3FFFF
bySA	Source Address of message sender	0 ... 253
BytePos	Starting position of bytes of data to be received	1 ... n
BitPos	Starting position of bits of data to be received	1 ... 8
DataType	Data type of data to be received	1 ... 3, 10 ... 16
DataLength	Volume of data for the global variable <code>VarAddr</code>	
VarAddr	Global variable into which the received value is entered	
TJ1939Rx	Control structure	
EventTime	Time lag between two telegrams (> Inhibit Time)	Default value: 1,000 ms
InhibitTime	Minimum time lag between two telegrams received (< EventTime)	Default value: 100 ms

### Return value

This function transmits the following return values to the higher-level program.

**Return value**

0	OK
-1	Error when checking parameters

**CANNo parameter**

This parameter specifies the number of the SAEJ1939 interface. CANNo = 0/1 is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

**Data Type parameter**

All allowed data types are listed below:

Byte types	Bit types	SAEJ1939
1	-	SAEJ1939_UNSIGNED8 SAEJ1939_BYTE
2	-	SAEJ1939_UNSIGNED16 SAEJ1939_WORD
4	-	SAEJ1939_UNSIGNED32 SAEJ1939_DWORD
n	-	SAEJ1939_STRING
-	1	SAEJ1939_1BIT
-	2	SAEJ1939_2BIT
-	3	SAEJ1939_3BIT
-	4	SAEJ1939_4BIT
-	5	SAEJ1939_5BIT
-	6	SAEJ1939_6BIT
-	7	SAEJ1939_7BIT

**Control structure  
TJ1939Rx**

```
TJ1939Rx : Struct
// Status of received message
    byStatus      : Byte;
// Priority of received message
    byPriority     : Byte;
End_Struct;
```



**How to apply this function**

```
Result := SAEJ1939AddRx (
    1,
    0xFEEE,
    0x00,
    2
    0
    SAEJ1939_BYTE,
    sizeof(var_Fueltemp),
    var_Fueltemp,
    struct_TJ1939Rx_EngineTemperatureTbl,
    1500,
    120);
```

**JetSym STX program**

The device with the SA of 20 wants to receive and display the current fuel temperature. The parameters **InhibitTime** and **EventTime** are not explicitly specified when calling up the function. In this case, the default values are used. The controller acquiring the fuel temperature has got SA 0. In practice, the address of the controller can be found in the engine manufacturer's documentation.

The fuel temperature has the SPN 174 and is a component (byte 2) of the PGN 65262 Engine Temperature 1.

```
Var
    bySAEJ1939Channel    : Byte;
    own_Source_Address   : Byte;

// PGN 65262 Engine Temperature 1
    Fueltemp             : Byte;
    EngineTemperatureTbl : J1939Rx;
End_Var;

Task main autorun

// Initializing CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

// Receiving the fuel temperature value
SAEJ1939AddRx (bySAEJ1939Channel, 65262, 0x00, 2, 1, SAEJ1939_BYTE,
    sizeof(Fueltemp), Fueltemp, EngineTemperatureTbl);

End_Task;
```

**Engine manufacturer's manual**

For information on the data (priority, PGN, SA and data byte structure) refer to the manual provided by the engine manufacturer.

## STX function SAEJ1939AddTx()

### Introduction

Calling the `SAEJ1939AddTx()` function prompts the device to send a specific message cyclically via the bus.

Cyclic sending continues until the `SAEJ1939Init()` function is called again.

The data is sent after the event time has elapsed. The inhibit time is not yet taken into account.

### Function declaration

```
Function SAEJ1939AddTx (
    CANNNo: Int,
    IPGN: Long,
    BytePos: Int,
    BitPos: Int,
    dataType: Int,
    DataLength: Int,
    const ref VarAddr,
    ref stJ1939: J1939Tx
    EventTime: Int,
    InhibitTime: Int,
) : Int;
```

### Function parameters

The `SAEJ1939AddTx()` function has the following parameters:

Parameter	Description	Value
CANNNo	CAN channel number	0/1 ... CANMAX
IPGN	PGN Parameter Group Number	0 ... 0x3FFFF
BytePos	Starting position of the byte of data to be sent	1 ... n
BitPos	Starting position of the bit of data to be sent	1 ... 8
dataType	Data type of data to be sent	1 ... 3, 10 ... 16
DataLength	Volume of data for the global variable <code>VarAddr</code>	
VarAddr	Global variable into which the value to be sent is entered	
J1939Tx	Control structure	
EventTime	Time lag between two telegrams (> Inhibit Time)	Default value: 1,000 ms
InhibitTime	Minimum time lag between two telegrams received (< EventTime)	Default value: 100 ms

**Return value**

This function transmits the following return values to the higher-level program.

**Return value**

0	OK
-1	Error when checking parameters

**CANNo parameter**

This parameter specifies the number of the SAEJ1939 interface. CANNo = 0/1 is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

**Data Type parameter**

All allowed data types are listed below:

Byte types	Bit types	SAEJ1939
1	-	SAEJ1939_UNSIGNED8 SAEJ1939_BYTE
2	-	SAEJ1939_UNSIGNED16 SAEJ1939_WORD
4	-	SAEJ1939_UNSIGNED32 SAEJ1939_DWORD
n	-	SAEJ1939_STRING
-	1	SAEJ1939_1BIT
-	2	SAEJ1939_2BIT
-	3	SAEJ1939_3BIT
-	4	SAEJ1939_4BIT
-	5	SAEJ1939_5BIT
-	6	SAEJ1939_6BIT
-	7	SAEJ1939_7BIT

**Control structure  
TJ1939Tx**

```
J1939Tx: Struct
// Status of sent message
    byStatus      : Byte;
// Priority of sent message
    byPriority     : Byte;
End_Struct;
```

**How to apply this function**

```
Result := SAEJ1939AddTx (
    1,
    0xFEEE,
    0x00,
    2
    0
    SAEJ1939_BYTE,
    sizeof(var_Fueltemp),
```

```
var_Fueltemp,  
struct_TJ1939Tx_EngineTemperatureTbl,  
1500,  
120);
```

---

### JetSym STX program

Redefining the priority.

Priority value 0 has the highest priority, priority value 7 has the lowest priority. A message with priority 6 can be superseded by a message with priority 4 (if the messages are sent at the same time). The parameters **InhibitTime** and **EventTime** are not explicitly specified when calling up the function. In this case, the default values are used.

Var

```
bySAEJ1939Channel   : Byte;  
own_Source_Address : Byte;  
  
// PGN 65262 Engine Temperature 1  
Fueltemp           : Byte;  
EngineTemperatureTbl : J1939Rx;
```

End\_Var;

Task main autorun

```
// Initializing CAN 1  
bySAEJ1939Channel := 1;  
own_Source_Address := 20;  
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);  
  
// PGN 65262 Engine Temperature  
// Setting a new priority  
EngineTemperatureTbl.byPriority := 6;  
SAEJ1939AddTx (bySAEJ1939Channel, 65262, 2, 1, SAEJ1939_BYTE,  
sizeof(Fueltemp), Fueltemp, EngineTemperatureTbl);
```

End\_Task;

---

### TSC1 Message

A special handling has to be considered for the TSC1 message (PGN = 0 BytePos = 8). This message contains a counter and a checksum.

If you want to send this message to the bus, it must be registered. The content of the current value on the bus is not transferred to the STX variable. Therefore, the value is not available to the application.

#### Example:

```
stateSAEJ1939Tx.byPriority = 3;  
SAEJ1939AddTx(1, 0x0000, 8, 1, SAEJ1939_BYTE, 1, value,  
stateSAEJ1939Tx, 10, 10);
```

---

### Engine manufacturer's manual

For information on the data (priority, PGN, SA and data byte structure) refer to the manual provided by the engine manufacturer.

---

## STX function SAEJ1939RequestPGN()

### Introduction

By calling `SAEJ1939RequestPGN()` the function a request is sent to the DA (Destination Address) after a PGN.

This function is terminated only if a valid value has been received or the timeout of 1,250 ms has elapsed.

To receive the value of the requested message, it must be registered for reception with the `SAEJ1939AddrX()` function.

This function must constantly be recalled in cycles.

### Function declaration

```
Function SAEJ1939RequestPGN(
    CANNo: Int,
    byDA: Byte,
    ulPGN: Long,
    byPriority: Byte,
) : Int;
```

### Function parameters

The `SAEJ1939RequestPGN()` function has the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	0/1 ... CANMAX
byDA	Destination Address Address from which the message is requested	0 ... 253 The own SA cannot be used
ulPGN	PGN Parameter Group Number	0 ... 0x3FFFF
byPriority	Priority	0 ... 7 Default value: 6

### Return value

This function transmits the following return values to the higher-level program.

#### Return value

0	Message has been received
-1	Timeout, as no reply has been received

### CANNo parameter

This parameter specifies the number of the SAEJ1939 interface. CANNo = 0/1 is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

### Data Type parameter

All allowed data types are listed below:

Byte types	Bit types	SAEJ1939
1	-	SAEJ1939_UNSIGNED8 SAEJ1939_BYTE
2	-	SAEJ1939_UNSIGNED16 SAEJ1939_WORD
4	-	SAEJ1939_UNSIGNED32 SAEJ1939_DWORD
n	-	SAEJ1939_STRING
-	1	SAEJ1939_1BIT
-	2	SAEJ1939_2BIT
-	3	SAEJ1939_3BIT
-	4	SAEJ1939_4BIT
-	5	SAEJ1939_5BIT
-	6	SAEJ1939_6BIT
-	7	SAEJ1939_7BIT

### How to apply this function

```
Result := SAEJ1939RequestPGN (  
    1,  
    0x00,  
    0xFEE5,  
    5);
```

### JetSym STX program

The device with its own SA of 20 wants to request PGN 65253 *Engine Hours* from an engine controller with SA 0. The SPN 247 *Engine Total Hours of Operation* is to be read from this PGN. It is therefore necessary to register reception of SPN 247 by calling the function `SAEJ1939AddRx()`.

The parameter "byPriority" is not explicitly specified when calling up the function. In this case, the default value is used.

```
Var  
    bySAEJ1939Channel    : Byte;  
    own_Source_Address  : Byte;  
  
    // PGN 65253 Engine Hours, Revolutions  
    EngineTotalHours    : Int;  
    EngineHoursTbl      : J1939Rx;  
End_Var;
```

```
Task main autorun
Var
    Return_value : Int;
End_Var;

// Initializing CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

// Engine Hours, Revolutions -- on Request
SAEJ1939AddRx (bySAEJ1939Channel, 65253, 0x00, 1, 0,
SAEJ1939_DWORD, sizeof(EngineTotalHours), EngineTotalHours, En-
gineHoursTbl, 5000, 150);

loop
    // Requesting total machine operating hours
    Return_value := SAEJ1939RequestPGN (bySAEJ1939Channel, 0x00,
65253);

    If Return_value Then
        Trace ('PGN Request failed');
    End_If;
    Delay(T#5s);
end_loop;
End_Task;
```

---

## STX function SAEJ1939GetDM1()

### Introduction

Calling up the function `SAEJ1939GetDM1()` requests the current diagnostics error codes (also see SAE J1939-73 No. 5.7.1). The corresponding PGN number is 65226. This function must constantly be recalled in cycles.

### Function declaration

```
Function SAEJ1939GetDM1 (
    CANNo: Int,
    bySA: Byte,
    ref stJ1939DM1stat: TJ1939DM1STAT
    ref stJ1939DM1msg: TJ1939DM1MSG
) : Int;
```

### Function parameters

The `SAEJ1939GetDM1()` function has the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	0/1 ... CANMAX
bySA	Source Address of message sender	0 ... 253 The own SA cannot be used
stJ1939DM1stat	IStatus IMsgCnt  IBuffer	Lamp Status Number of received messages Size of variable stJ1939DM1msg
stJ1939DM1msg	ISPN byOC byFMI	Error code Error counter Error type

### Return value

This function transmits the following return values to the higher-level program.

#### Return value

0	OK
-1	Error when checking parameters

### CANNo parameter

This parameter specifies the number of the SAEJ1939 interface. `CANNo = 0/1` is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.



**stJ1939DM1stat.IStatus**      **Default: 0xFF00**

Type	Byte	Bit group	Description
Status	1	8 - 7	Malfunction Indicator Lamp Status
		6 - 5	Red Stop Lamp Status
		4 - 3	Amber Warning Lamp Status
		2 - 1	Protect Lamp Status
Flash	2	8 - 7	Flash Malfunction Indicator Lamp
		6 - 5	Flash Red Stop Lamp
		4 - 3	Flash Amber Warning Lamp
		2 - 1	Flash Protect Lamp

Type	Byte	Bit group Value	Description
Status	1	00	Lamps off
		01	Lamps on
Flash	2	00	Slow Flash (1 Hz, 50 % duty cycle)
		01	Fast Flash (2 Hz or faster, 50 % duty cycle)
		10	Reserved
		11	Unavailable / Do Not Flash

**stJ1939DM1msg**

**Default value:**

ISPN = 0

byOC = 0

byFMI = 0

For older controllers (grandfathered setting):

ISPN = 524287 (0x7FFFF)

byOC = 31 (0x1F)

byFMI = 127 (0x7F)

**How to apply this function**

```
Result := SAEJ1939GetDM1 (
    1,
    0x00,
    stdmlstat_pow,
    stdmlmsg_pow,);
```

### JetSym STX program

By calling the SAEJ1939GetDM1 () function, the device requests the current diagnostics error codes (PGN 65226).

```
Var
    bySAEJ1939Channel    : Byte;
    own_Source_Address   : Byte;
    stdmlstat_pow        : J1939DM1Stat;
    stdmlmsg_pow         : Array[10] of J1939DM1Msg;
    MyTimer              : Timer;
End_Var;

Task main autorun

// Initializing CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

TimerStart (MyTimer, T#2s);

Loop

When (TimerEnd (MyTimer)) Continue;

// Requesting diagnostics error codes DM1 POW
stdmlstat_pow.lBuffer := sizeof (stdmlmsg_pow);
SAEJ1939GetDM1 (bySAEJ1939Channel, 0x00, stdmlstat_pow,
stdmlmsg_pow);

TimerStart (MyTimer, T#2s);

End_Loop;

End_Task;
```

---

## STX function SAEJ1939GetDM2()

### Introduction

Calling the SAEJ1939GetDM2 () function requests the diagnostics error codes that preceded the current ones (also see SAEJ1939-73 No. 5.7.2). The corresponding PGN number is 65227.

### Function declaration

```
Function SAEJ1939GetDM2 (
    CANNNo: Int,
    bySA: Byte,
    ref stJ1939DM2stat: TJ1939DM2STAT
    ref stJ1939DM2msg: TJ1939DM2MSG
) : Int;
```

### Function parameters

The SAEJ1939GetDM2 () function has the following parameters:

Parameter	Description	Value
CANNNo	CAN channel number	0/1 ... CANMAX
bySA	Source Address of message sender	0 ... 253 The own SA cannot be used
stJ1939DM2stat	IStatus IMsgCnt  IBuffer	Lamp Status Number of received messages Size of variable stJ1939DM2msg
stJ1939DM2msg	ISPN byOC byFMI	Error code Error counter Error type

### Return value

This function transmits the following return values to the higher-level program.

#### Return value

0	OK
-1	Error when checking parameters

### CANNNo parameter

This parameter specifies the number of the SAEJ1939 interface. CANNNo = 0/1 is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

**stJ1939DM2stat.IStatus**      **Default:** 0xFF00

Type	Byte	Bit group	Description
Status	1	8 - 7	Malfunction Indicator Lamp Status
		6 - 5	Red Stop Lamp Status
		4 - 3	Amber Warning Lamp Status
		2 - 1	Protect Lamp Status
Flash	2	8 - 7	Flash Malfunction Indicator Lamp
		6 - 5	Flash Red Stop Lamp
		4 - 3	Flash Amber Warning Lamp
		2 - 1	Flash Protect Lamp

Type	Byte	Bit group Value	Description
Status	1	00	Lamps off
		01	Lamps on
Flash	2	00	Slow Flash (1 Hz, 50 % duty cycle)
		01	Fast Flash (2 Hz or faster, 50 % duty cycle)
		10	Reserved
		11	Unavailable / Do Not Flash

**stJ1939DM2msg**

**Default value:**

ISPN = 0

byOC = 0

byFMI = 0

For older controllers (grandfathered setting):

ISPN = 524287 (0x7FFFF)

byOC = 31 (0x1F)

byFMI = 127 (0x7F)

**How to apply this function**

```
Result := SAEJ1939GetDM2 (
    1,
    0x00,
    stdm2stat_pow,
    stdm2msg_pow,);
```

**JetSym STX program**

By calling the `SAEJ1939GetDM2()` function, the device requests the current diagnostics error codes (PGN 65227).

```
Var
    bySAEJ1939Channel    : Byte;
    own_Source_Address   : Byte;
    stdm2stat_pow        : J1939DM2Stat;
    stdm2msg_pow         : Array[10] of J1939DM2Msg;
End_Var;

Task main autorun

// Initializing CAN 1
bySAEJ1939Channel := 1;
own_Source_Address := 20;
SAEJ1939Init (bySAEJ1939Channel, own_Source_Address);

// Required for a cyclical task
TaskAllEnableCycle ();
EnableEvents;
End_Task;

Task t_RequestPGN_5000 cycle 5000

Var
    Return_value : Int;
End_Var;

// Requesting diagnostics error codes DM2 POW
stdm2stat_pow.lBuffer := sizeof (stdm2msg_pow);
Return_value := SAEJ1939GetDM2 (bySAEJ1939Channel, 0x00,
stdm2stat_pow, stdm2msg_pow);

If Return_value Then
    Trace ('DM2 Request failed');
End_If;

End_Task;
```

## STX function SAEJ1939SetSPNConversion()

### Introduction

Calling the `SAEJ1939SetSPNConversion()` function determines the arrangement of bytes in the message which is requested by the `SAEJ1939GetDM1()` or `SAEJ1939GetDM2()` function. In other words, this function lets you specify the conversion method.

### Function declaration

```
Function SAEJ1939SetSPNConversion(
    CANNo: Int,
    bySA: Byte,
    iConversionMethod: Int,
) : Int;
```

### Function parameters

The `SAEJ1939SetSPNConversion()` function has the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	0/1 ... CANMAX
bySA	Source Address of message sender	0 ... 253
iConversionMethod	Conversion method	1 ... 4 4: Automatic detection 2: Default

### Return value

This function transmits the following return values to the higher-level program.

#### Return value

0	OK
-1	Error when checking parameters

### CANNo parameter

This parameter specifies the number of the SAEJ1939 interface. `CANNo = 0/1` is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (`CANMAX`) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

### How to apply this function

```
Result := SAEJ1939SetSPNConversion(
    1,
    0xAE,
    4);
```

## STX function SAEJ1939GetSPNConversion()

### Introduction

Calling the `SAEJ1939GetSPNConversion()` function determines the currently set conversion method.

### Function declaration

```
Function SAEJ1939SetSPNConversion (
    CANNo: Int,
    bySA: Byte,
    iConversionMethod: Int,
) : Int;
```

### Function parameters

The `SAEJ1939GetSPNConversion()` function has the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	0/1 ... CANMAX
bySA	Source Address of message sender	0 ... 253
iConversionMethod	Conversion method	1 ... 4 4: Automatic detection 2: Default

### Return value

This function transmits the following return values to the higher-level program.

#### Return value

0	OK
-1	Error when checking parameters

### CANNo parameter

This parameter specifies the number of the SAEJ1939 interface. CANNo = 0/1 is assigned to the first interface. The number of SAEJ1939 interfaces depends on the device. For information on the maximum number of SAEJ1939 interfaces (CANMAX) refer to the chapters *Technical Specifications* and *Quick Reference* in the corresponding user manual.

### How to apply this function

```
Result := SAEJ1939GetSPNConversion (
    1,
    0xAE,
    actual_conversion_method);
```

Jetter AG  
Graeterstrasse 2  
71642 Ludwigsburg | Germany

Phone +49 7141 2550-0  
Fax +49 7141 2550-425  
[info@jetter.de](mailto:info@jetter.de)  
[www.jetter.de](http://www.jetter.de)

We automate your success.