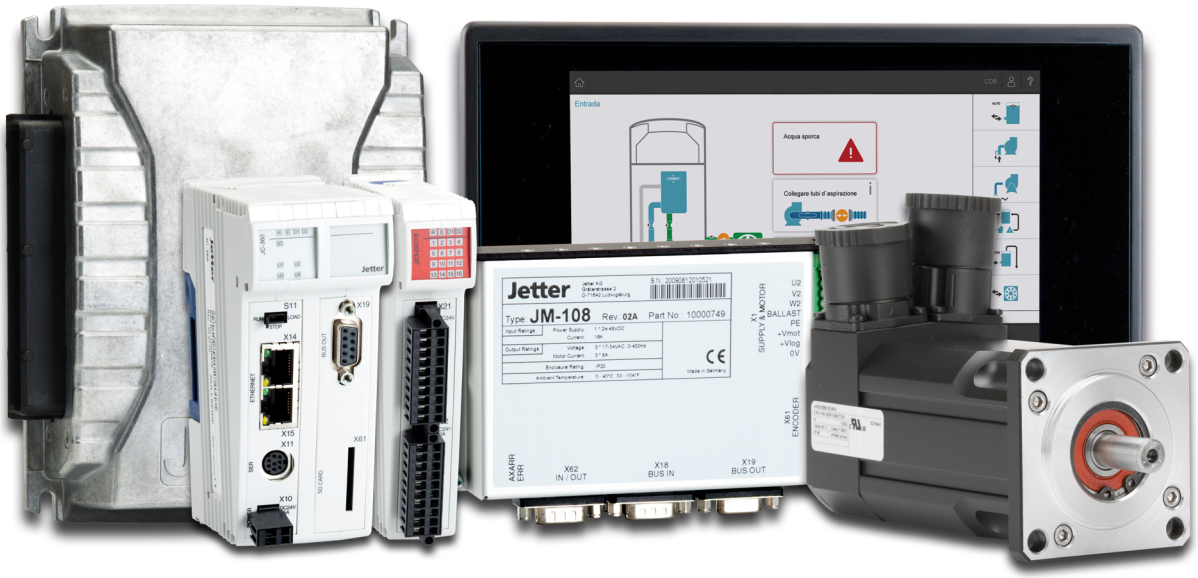
Jetter automation



Application-Oriented Manual

SAE J1939 STX API

60881084

We automate your success.

Item # 60881084
Revision 1.01
April 2017 / Printed in Germany

Table of Contents

1 SAE J1939

Introduction

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

Contents

J1939 protocol - Layer description

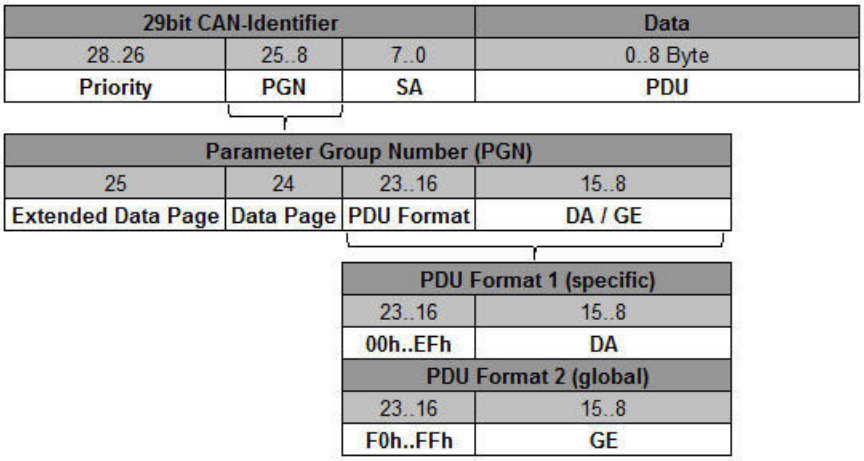
Physical layer

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):
0x18**FEE9***27*

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 byte
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 byte	Engine Coolant Temperature	110
2	1 byte	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 byte	Engine Intercooler Temperature	52
8	1 byte	Engine Intercooler Thermostat Opening	1134

2 SAE J1939 STX API

Introduction

This chapter describes the STX functions of the SAE J1939 STX API.

The SAE J1939 Standard

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.

Application

These STX functions are used in communication between the controllers apt for SAE J1939 and other ECUs in the vehicle. As a rule, engine data, such as RPM, speed or coolant temperature are read and displayed.

Documentation

The key SAE J1939 specifications are:

- J1939-11 - Information on the physical layer
- J1939-21 - Information on the data link layer
- J1939-71 - Information on the application layer vehicles
- J1939-73 - Information on the application layer range analysis
- J1939-81 - Network management

Devices

The following devices have got the SAE J1939-STX-API feature:

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

Contents

STX function SAEJ1939Init()

Introduction

Calling up the `SAEJ1939Init()` function initializes one of the CAN busses (not CAN 0 as this is reserved for CANopen®) for use with the J1939 protocol. From then on, the device has got 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(
    CANNo:Int,
    mySA:Byte,
) :Int;
```

Function parameters

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

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

Return value

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

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

CANNo parameter

This parameter specifies the number of the SAEJ1939 interface. CANNo = 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 manual.

How to use this function

Initializing CAN bus 1. The device has got Node 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 SAEJ1939SetSA()

Introduction

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

Function declaration

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

Function parameters

The function SAEJ1939SetSA() comprises the following parameters:

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

Return value

This function transfers 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 = 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 manual.

How to use 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 function `SAEJ1939GetSA()` lets you determine your own SA (Source Address).

Function declaration

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

Function parameters

The function `SAEJ1939GetSA()` comprises the following parameters:

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

Return value

This function transfers 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 = 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 manual.

How to use this function

This function returns the currently set SA.

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

STX function SAEJ1939AddRx()

Introduction

Calling up the function `SAEJ1939AddRx()` prompts 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. Cyclical reading continues until the function `SAEJ1939Init()` is called up again.

Function declaration

```
Function SAEJ1939AddRx(
    CANNo:Int,
    IPGN:Long,
    bySA:Byte,
    BytePos:Int,
    BitPos:Int,
    DataType:Int,
    DataLength:Int,
    const ref VarAddr,
    ref stJ1939:TJ1939Rx
    EventTime: Int,
    InhibitTime: Int,
) :Int;
```

Function parameters

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

Parameter	Description	Value
CANNo	CAN channel number	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 VarAddr	
VarAddr	Global variable into which the received value is entered	
TJ1939Rx	Control structure	
EventTime	Time lag between two telegrams (> InhibitTime)	Default value: 1,000 ms
InhibitTime	Minimum time lag between two telegrams received (< EventTime)	Default value: 100 ms

Return value

This function transfers 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 = 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 manual.

DataType parameter

All allowed data types are listed below:

Byte types	Bit types	
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 use 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 own 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 for capturing 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.

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel    : Byte;
    own_Source_Address   : Byte;

// PGN 65262 Engine Temperature 1
    Fueltemp              : Byte;
    EngineTemperatureTbl  : TJ1939Rx;
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 up the function `SAEJ1939AddTx()` prompts the device to cyclically send a specific message via the bus.

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

Date are sent once the event time has elapsed. Up to now, the inhibit time has not been considered yet.

Function declaration

```
Function SAEJ1939AddTx(
    CANNo:Int,
    IPGN:Long,
    BytePos:Int,
    BitPos:Int,
    dataType:Int,
    DataLength:Int,
    const ref VarAddr,
    ref stJ1939:TJ1939Tx
    EventTime: Int,
    InhibitTime: Int,
) :Int;
```

Function parameters

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

Parameter	Description	Value
CANNo	CAN channel number	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 VarAddr	
VarAddr	Global variable into which the value to be sent is entered	
TJ1939Tx	Control structure	
EventTime	Time lag between two telegrams (> InhibitTime)	Default value: 1,000 ms
InhibitTime	Minimum time lag between two telegrams received (< EventTime)	Default value: 100 ms

Return value

This function transfers 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 = 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 manual.

DataType parameter

All allowed data types are listed below:

Byte types	Bit types	
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

```
TJ1939Tx : Struct
// Status of sent message
            byStatus      : Byte;
// Priority of sent message
            byPriority    : Byte;
          End_Struct;
```

How to use 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 explicity specified when calling up the function. In this case, the default values are used.

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel   : Byte;
    own_Source_Address  : Byte;

// PGN 65262 Engine Temperature 1
    Fueltemp              : Byte;
    EngineTemperatureTbl  : TJ1939Rx;
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, 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 SAEJ1939RequestPGN()

Introduction

Calling up the function `SAEJ1939RequestPGN()` sends a request to the DA (Destination Address) following a PGN.

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

To obtain the value of the requested message its receipt must be scheduled using the function `SAEJ1939AddRx ()`.

This function must constantly be recalled in cycles.

Function declaration

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

Function parameters

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

Parameter	Description	Value
CANNo	CAN channel number	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 transfers 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 = 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 manual.

DataType parameter

All allowed data types are listed below:

Byte types	Bit types	
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 use this function

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

JetSym STX program

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

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

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel   : Byte;
    own_Source_Address  : Byte;

// PGN 65253 Engine Hours, Revolutions
    EngineTotalHours    : Int;
    EngineHoursTbl      : TJ1939Rx;
End_Var;
```

```
Task main autorun

// 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,
EngineHoursTbl, 5000, 150);

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

End_Task;


Task t_RequestPGN_5000 cycle 5000

Var
    Return_value  : Int;
End_Var;

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

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

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 function `SAEJ1939GetDM1()` comprises the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	1 ... CANMAX
bySA	Source Address of message sender	0 ... 253 The own SA cannot be used
stJ1939DM1stat	lStatus lMsgCnt lBuffer	Lamp status Number of received messages Size of variable stJ1939DM1msg
stJ1939DM1msg	lSPN byOC byFMI	Error code Error counter Error type

Return value

This function transfers 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 = 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 manual.

stJ1939DM1stat.lStatus **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 use this function

```
Result := SAEJ1939GetDM1(
    1,
    0x00,
    stdm1stat_pow,
    stdm1msg_pow,);
```

JetSym STX program

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

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel   : Byte;
    own_Source_Address  : Byte;
    stdm1stat_pow       : TJ1939DM1STAT;
    stdm1msg_pow        : Array[10] of STJ1939DM1MSG;
    MyTimer             : TTimer;
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
stdm1stat_pow.lBuffer := sizeof (stdm1msg_pow);
SAEJ1939GetDM1 (bySAEJ1939Channel, 0x00, stdm1stat_pow,
stdm1msg_pow);

TimerStart (MyTimer, T#2s);

End_Loop;

End_Task;
```

STX function SAEJ1939GetDM2()

Introduction

Calling up the function `SAEJ1939GetDM2()` requests the diagnostics error codes that preceded the current ones (also see SAE J1939-73 No. 5.7.2). The corresponding PGN number is 65227.

Function declaration

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

Function parameters

The function `SAEJ1939GetDM2()` comprises the following parameters:

Parameter	Description	Value
CANNo	CAN channel number	1 ... CANMAX
bySA	Source Address of message sender	0 ... 253 The own SA cannot be used
stJ1939DM2stat	lStatus lMsgCnt lBuffer	Lamp status Number of received messages Size of variable stJ1939DM2msg
stJ1939DM2msg	lSPN byOC byFMI	Error code Error counter Error type

Return value

This function transfers 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 = 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 manual.

stJ1939DM2stat.lStatus **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 use this function

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

JetSym STX program

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

```
#Include "SAEJ1939.stxp"

Var
    bySAEJ1939Channel    : Byte;
    own_Source_Address   : Byte;
    stdm2stat_pow        : TJ1939DM2STAT;
    stdm2msg_pow         : Array[10] of STJ1939DM2MSG;
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 up the function `SAEJ1939SetSPNConversion()` determines the configuration of bytes in the message, which is requested using function `SAEJ1939GetDM1()` or `SAEJ1939GetDM2()`. 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 function `SAEJ1939SetSPNConversion()` comprises the following parameters:

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

Return value

This function transfers 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 = 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 manual.

How to use this function

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

STX function SAEJ1939GetSPNConversion()

Introduction

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

Function declaration

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

Function parameters

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

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

Return value

This function transfers 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 = 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 manual.

How to use this function

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

Jetter automation

Jetter AG
Graeterstrasse 2
71642 Ludwigsburg | Germany

Phone +49 7141 2550-0
Fax +49 7141 2550-425
info@jetter.de
www.jetter.de

We automate your success.