EI-000022-D

SAE J1939 PRESSURE TRANSDUCER

TABLE OF CONTENTS

TYPE: SMC

PROTOCOL DESCRIPTION

Edited by Cornelius Schneider

1. GENERAL

The protocol SAE J1939 is based on CAN.

The documents SAE J1939/1x describe its structure; Chapter 3 contains a general description.

The pressure transducers produced by ADZ NAGANO GmbH are of the ECU-type I. Thus, they do not include bus termination resistors.

2. ABBREVIATION INDEX

CAN	Controller Area Network
DA	Destination Address
ECU	Electronic Control Unit
PDU	Protocol Data Unit
PGN	Parameter Group Number
SA	Source Address
SAE	Society of Automotive Engineers
SPN	Suspect Parameter Group

3. CAN BUS TOPOLOGY

The CAN bus is a linear structure (see Figure 1). All data are transferred in differential mode using two electrical lines (CAN high, CAN low). On both of its ends, the bus requires a termination network (each resistor, 120Ω) to provide a defined idle state.

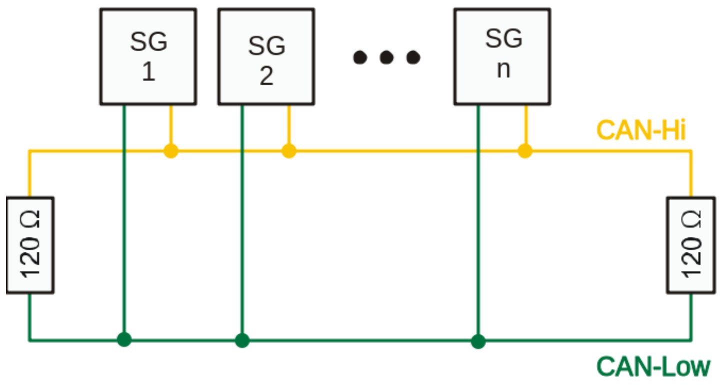


Figure 1:
Linear CAN bus topology with terminating resistors [1]

For a limited extent, bus stubs may be used for realizing a star topology. Be aware that in this case, the terminating resistors have to be adjusted.

In combination with the protocol SAE J1939, the bus nodes are called ECUs.

4. BIT RATE

The default bit rate is set to 125kBit/s. For special applications, different bit rates can be realized.

5. NETWORK MANAGEMENT

The network management for SAE-J1939 is described in the document "J1939/81". In default settings state, the pressure transducer fulfils the minimal requirements for "resolving address conflicts" and "checking for multiple device addresses". Furthermore, it supports the automatic address alteration during runtime.

The fields for a clear determination of the devices' functions may be assigned according to the customer's needs.

6. CAN MESSAGE FORMAT

Figure 2 depicts the CAN message format when using SAE J1939. The CAN-ID contains message priority, PGN and source address.

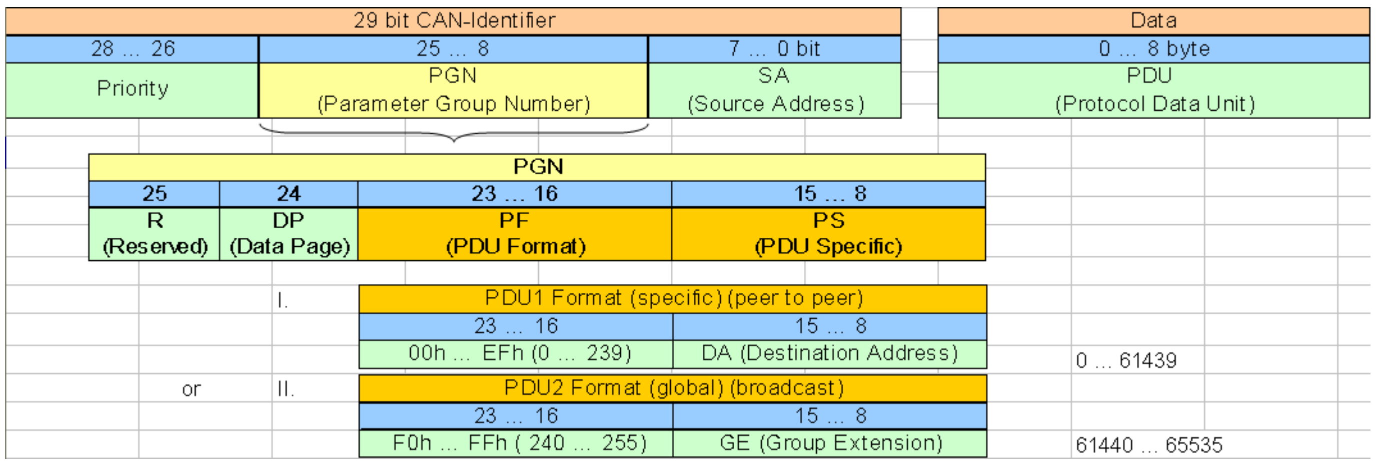


Figure 2: SAE J1939 CAN message format

6.1 Parameter Group Number (PGN)

The Parameter Group Number (PGN) consists of the fields "R" (Reserved), "DP" (Data Page), "PF" (PDU Format) and "PS" (PDU Specific). The first two fields ("R" and "DP") are not used within the default parameter set and both do contain zeros. Document SAE J1939/71 defines various PGNs. Each of them has specifically structured payload and other parameters (e.g. repetition time).

The usage of proprietary data formats is possible, as long as the field "PF" is assigned the values 239 (PGN Proprietary A) or 255 (PGN Proprietary B).

6.1.1 PGN Proprietary A

This PGN is used when transferring data in a targeted manner. The field "PS" declares the destination address.

6.1.2 PGN Proprietary B

Broadcast messages use the PGN Proprietary B. In this case, the field "PS" declares supporting payload data formats.

7. USER DATA FORMATS

The user data format must be interpreted according to the used PGN. A PGN may contain one or more SPNs (*Suspect Parameter Numbers*). The SPN data structure is described by SAE J1939/71.
ADZ NAGANO GmbH pressure transducers by default do use a proprietary data format.

7.1 Interpreting the transmitted pressure values

Each message's data field length is 8 bytes. The first two bytes contain the pressure value; the following bytes are assigned 255 (0xFF). The MSB (*Most Significant Byte*) will be sent at first (*known as Big Endian*).
Received data need to be multiplied with the *Resolution*. Subsequently, the *Offset* is subtracted. *Both Resolution* and *Offset* can be found in the pressure transducer's datasheet.

Example:

Pressure transducer:	-1...10bar
Resolution:	0,01 bar / bit
Offset:	1 bar
Pressure:	9,00 bar
Data bytes (HEX notation):	0x03 0xE8 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
Relevant data bytes:	0x03 E8 = $1000_{decimal}$

- Calculation:

$$\begin{aligned} Pressure &= Pressure_{Digital} \cdot Resolution - Offset \\ &= 1000 \cdot 0{,}01 \frac{bar}{bit} - 1\ bar \\ &= 9 bar \end{aligned}$$

8. THE DEFAULT SETTINGS SET

The following table shows a default configuration. Specific configurations and settings may be found in part-number depending product datasheets.

Parameter	Default value	Meaning
Transmission baud rate	125 kBit/s	Transmission speed.
Start address	208 *(0xD0)*	Assigned address at boot up time.
PGN	65330 *(0xFF 32)*	Parameter number of data.
Priority	7	Message priority (0 = highest).
Transmission cycle time (also repetition time)	100ms	Repetition time of the pressure messages.
SPN	According to pressure range.	Describes data interpretation.
Manufacturer ID	455 *(0x01 C7)*	Manufacturer ID of ADZ NAGANO GmbH
ECU Instance	1	These fields hold the device's function within the system.
Function Instance	3	
Function	4	
Industry Group	5	
Vehicle System	7	
Vehicle System Instance	1	
Arbitrary Address Capable	1	Automatic address obtaining enabled.

9. SETTINGS PGN

9.1 General information

The *Settings PGN* can be used to read and write registers from the *SMC pressure/temperature transducer*. These registers may contain settings or status/informational data, depending on the address that is being accessed.

There are two types of accesses: reading and writing accesses. While writing accesses must be unlocked before using (refer to section 9.1.2), read accesses can always be applied.

A typical *Settings PGN* access consists of three important things:

- The *CAN-ID* holding the address of the *J1939* device that is being accessed,
- *Register Access Code* (containing one read/write bit and the 15-bit register address; discussed below),
- In case of write access: The *payload data* that shall be written (up to six bytes). The amount of *payload data bytes* (and thus the *DLC*) depends on the register being written to. Important: All multi-byte values need to be ordered as *Little Endian*!

The figure below shows how to assemble the necessary **CAN-ID**:

MSB			LSB
5 bits	**8 bits**	**8 bits**	**8 bits**
Info field. Typically "0x18"	*Must be "0xEF"*	*Destination Address (see below)*	*Source Address*

The field *Destination Address* contains the address of the device being accessed. When choosing the value 0xFF combined with subsequent *register write accesses*, all present *SMC pressure/temperature transducer* devices will be configured.

The 16-bit **Register Access Code** consists (as already mentioned above) of two fields:

1) A one-bit read/write value located at the *most significant bit*. The value '1' stands for writing; the value '0' stands for reading access.
2) The 15-bit address of the register being accessed. An overview of all available registers can be found at section 9.2.

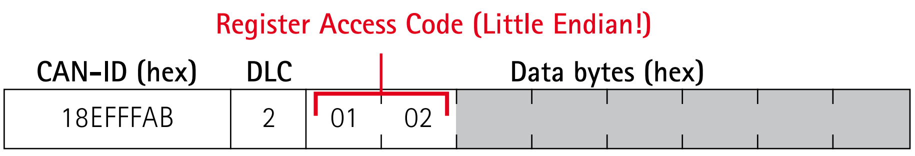
Two more important things:

- Before writing values to any register, the **write access must be unlocked**. In order to unlock write access, refer to section 9.1.2.

- Data written to any registers will not automatically be applied to *non-volatile EEPROM*. Thus, the **save process** explicitly must be performed **after applying configuration values**.The save procedure is described in section 9.2.2.1.

9.1.1 Exemplary register read access

The following figure shows how to build up a CAN message containing a valid read access.



Exemplary access for reading a value from register 0x0201.

A possible response to this access could be as follows:

Register Access Code
Read value (in this case int32; Little Endian!)

CAN-ID (hex)	DLC	Data bytes (hex)							
18EFABxx	2	01	02	DD	CC	BB	AA		

Exemplary access for reading a value from register 0x0201.

xx stands for the trancducer's address

The amount of data bytes contained in the response depends on the register that was accessed. Refer to the overview of available registers, located at section 9.2.

9.1.2 Unlocking the register write access

Before writing to any registers, the write access must be unlocked. The following CAN message holds the necessary data in order to unlock *all present ADZ J1939 transducers*.

Unlock Code

CAN-ID (hex)	DLC	Data bytes (hex)							
18EFFFAB	8	41	42	5A	2B	50	4D	53	54

Unlocking register write access for all present ADZ j1939 transducers.

After transferring the message above, each present ADZ J1939 transducer sends out the message below.

Unlock Code

CAN-ID (hex)	DLC	Data bytes (hex)							
18EFFFAB	8	41	42	5A	2B	50	4D	53	54

Each trancduser's response after successfully unlocking.

xx stands for the trancducer's address

After performing this procedure, all present transducers' register write access will be unlocked until reboot. Alternatively, the access can be manually locked by writing to register 0x0000.

9.1.3 Exemplary register write access

Requirement: The register write access has already been unlocked (refer to 9.1.2).

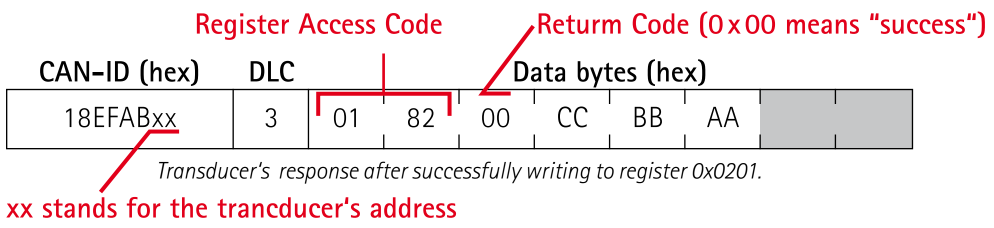
The amount of data bytes (and data values) depends on the register that is being accessed. The following CAN message shows how to write exemplary data contents to register 0x0201.

Register Access Code (Little Endian: MSB is '1')
Data value (in thus case int32; Little Endian)

CAN-ID (hex)	DLC	Data bytes (hex)							
18EFFFAB	6	01	82	DD	CC	BB	AA		

Exemplary write access to register 0x0201.

In case the data values were assembled correctly and write access is unlocked, the transducer transfers the following message.


Register Access Code
Returm Code (0x00 means "success")
xx stands for the trancducer's address

CAN-ID (hex)	DLC	Data bytes (hex)							
18EFABxx	3	01	82	00	CC	BB	AA	▨	▨

Transducer's response after successfully writing to register 0x0201.

After performing all necessary write accesses, the user might want to save the applied settings to non-volatile EEPROM. Section 9.2.2.1 shows how to do so.

9.2 REGISTER OVERVIEW

The following sections describe the accessible entries of the *SMC pressure/temperature transducer's* register set.

15-bit Register address (HEX)	Allowed access types	Description
General CAN registers		
0x0000	W	Locks all further write accesses to any registers
0x0001	R	Transducers hardware revision.
0x0002	R	Transducers firmware revision.
0x0010	RW	Measurements' endianness
0x0020	RW	J1939 start address
0x0040	W	Bootup baudrate
0x0050	RW	Milliseconds timestamp – Upper 32 bits
0x0051	RW	Milliseconds timestamp – Lower 32 bits. Access at first!
General transducer registers		
0x0100	W	Save current settings to EEPROM
0x0110	W	Reset to factory settings
0x0120	W	Perform reboot
Primary measuring channel settings		
0x0200	RW	Gain (Float32)
0x0201	RW	Gain (Int32)
0x0210	RW	Offset (Float32)
0x0211	RW	Offset (Int32)
0x0220	W	Apply current readings as 'zero'
0x0221	RW	'Zero' value
Measurements-PGN 1 settings		
0x0800	RW	General PGN settings
0x0810	RW	Static mask 1
0x0811	RW	Static mask 2
0x0820	RW	Mapping entries 1
0x0821	RW	Mapping entries 2

9.2.1 General CAN registers

9.2.1.1 0x0001, 0x0002 (Transducer hardware / firmware revision)

These read-only registers hold the hardware / firmware revision (each Uint16). The returned decimal values are encoded in the following manner: ***YYWW*** where *YY* corresponds to the year, *WW* to the week.

9.2.1.2 0x0010 (Measurements' endianness)

This register changes the endianness of all measurement values.

Type of data value	Possible values	Notes
Uint8	0x00 ... 0x01	0x00: Little Endian 0x01: Big Endian

9.2.1.3 0x0020 (J1939 start address)

This register allows to access the current J1939 start address used for *address claiming process*.

The amount of data values differs when reading/writing – that's why both access types will be discussed separately.

Read access:

Type of data value	Possible values	Notes
Uint8	0x00 ... 0xFD	Contains the start address

Write access:

Type of data value	Possible values	Notes
Uint8	0x00 ... 0xFD	Contains the start address
Uint8	0x00 or 0x01	Makes (if 0x01) the transducer perform *address claiming* immediately after writing.

9.2.1.4 0x040 (Boot-up baud rate)

This register changes the baudrate at bootup. A change of this register do not change the baudrate immediately.

Type of data value	Possible values	Notes
Uint8	0x01 ... 0x08	0x01: 20 kBit/s 0x02: 50 kBit/s 0x03: 100 kBit/s 0x04: 125 kBit/s 0x05: 250 kBit/s 0x06: 500 kBit/s 0x07: 800 kBit/s 0x08: 1 MBit/s

9.2.1.5 0x0050 / 0x0051 (Milliseconds timestamp)

These registers are incremented every millisecond automatically and they can be mapped to a PGN. So it is possible to transmit timestamps and process values together. After power on the register will be initialized with 0. Register 0x0050 held the upper 32 bit of the 64bit timestamp.
With a write access it is possible to synchronize the timestamp to a higher ranking system.

NOTE: For read / write access to these registers it is highly recommended to access register 0x0051 at first.

9.2.2 General transducer registers

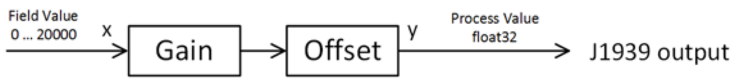
9.2.2.1 0x0001 (Save current settings to EEPROM)
An access to this register saves all settings to EEPROM and make them available through a reset of the transducer.

9.2.2.2 0x0110 (Reset to factory settings)
An access to this register reset all setting to the factory defaults.

9.2.2.3 0x0120 (Perform reboot)
An access to this register perform a reboot of the transducer. All changes which are not save will be lost.

9.2.3 Primary measuring channel settings

9.2.3.1 30 0x0200 ... 0x0219 (Gain and offset parameters)
These registers enable the user to manipulate the output values' scaling settings.
The following image shows the linear scaling procedure.



The transducer's nominal measuring range corresponds to the internal *Field Value* range of [0 ... 20000]. Using the *gain* and *offset* parameters, these internal values are being scaled according to the following formula:

$$y = Gain \cdot x + Offset \quad \text{with} \quad \begin{array}{l} x = Field\ Value \\ y = Process\ Value \end{array}$$

Both gain and offset parameters can be read and written as Int32 or Float32 data types.

9.2.3.2 0x0220 (Apply current reading as 'zero')
An access to this register start an offset compensation. The actual reading will be used as offset reading.

9.2.3.3 0x0221 ('zero' value)
This register held the actual value for the offset compensation.

Type of data value	Possible values	Notes
Int16	-32768..32767	Value for offset compensation

9.2.4 Measurements-PGN 1 settings
The following sub-sections explain all necessary registers enabling the user to configure measurements-PGN 1.

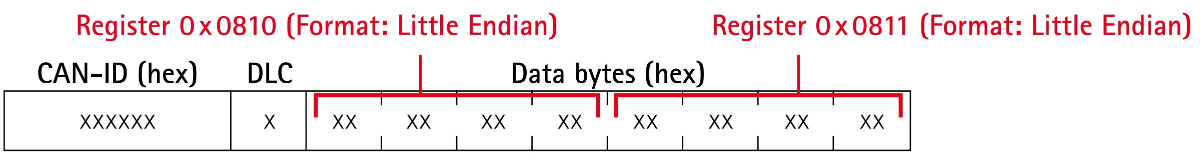
9.2.4.1 0x0800 (General PGN settings)
This register allows to access general settings of measurements-PGN 1. The register contents consist of 6 data bytes; the following table shows how to assemble the register contents.

Type of data value	Possible values	Notes
Uint8		PDU Format
Uint8		PDU Specific
Uint16		PGN transmission rate in milliseconds.

Type of data value	Possible values	Notes
Uint8		Split up 4+4 bits: • The **uppermost 4 bits** contains priority (possible values 0x0..0x7) • The **lowermost 4 bits** contain the PGN *Data Page ("DP")*. 0x0 DP = 0; 0x1 -> DP=1
Uint8		Splits up to 4+4 bits: • The **uppermost 4 bits** contain the flag *AutostartEnabled (0x8 = Enabled).* • The lowermost 4 bits contain the PGN *Data Length Code ("DLC").*

9.2.4.2 0x0810 ... 0x0811 (Static mask)

These register held values which are copied to the PGN message before a transmission is started. The intention is to applied static values to a message. The values are overwritten from the Mapping entries. Refer section 9.2.4.3 for an example.


Register 0x0810 (Format: Little Endian)
Register 0x0811 (Format: Little Endian)
CAN-ID (hex)
DLC
Data bytes (hex)
XXXXXX
X
XX
XX
XX
XX
XX
XX
XX
XX

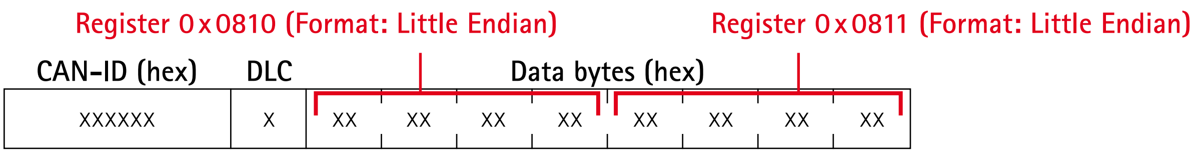
Assignment static mask bytes vs PGN - message

9.2.4.3 0x0820 ... 0x0821 (Mapping entries)

The intention of these register is to design the structure of the PGN which is send. The possible data and there identification number is listed in the table below.

value	Size	Usage
0	Uint8	Data of this Position is not overwritten
1	Uint8	Status of transducer
	Uint32	LSB of Timestamp in Millisecond
3	Uint64	Timestamp in millisecond
5	Uint32	ADZ – serial number without "Z"
6	Uint64	ADZ – serial number without "Z"
7	Uint8+Uint32	ADZ – serial number with "Z"
11	Uint32	user serial number
12	Uint64	user serial number
40	Int16	Field value primary channel
41	Int16	Process value primary channel
42	Int32	Process value primary channel
43	Float32	Process value primary channel
50	Int16	Field value primary channel
51	Int16	Process value secondary channel
52	Int32	Process value secondary channel
53	Float32	Process value secondary channel

The assignment between the registers and the PGN is shown below.


Register 0x0810 (Format: Little Endian)
Register 0x0811 (Format: Little Endian)
CAN-ID (hex)
DLC
Data bytes (hex)
xxxxxx
x
xx
xx
xx
xx
xx
xx
xx
xx

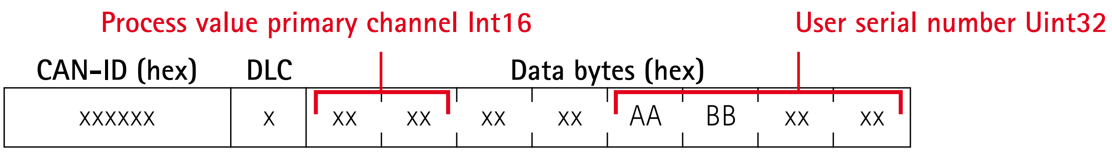
Assignment mapping bytes vs PGN - message

The internal order of assigning a PGN – message is:

- Allocating a buffer with 8 Byte
- Coping the static mask to the buffer
- Coping the values to the PGN correspondent the mapping registers. If the mapping has the value 0 the equivalent byte of the PGN is not changed.

Note: If the mapped data is greater than Uint8 the followed mapping Bytes must be set to 0 to prevent an overwriting of the Data

Example:


Process value primary channel Int16
User serial number Uint32
CAN-ID (hex)
DLC
Data bytes (hex)
xxxxxx
x
xx
xx
xx
xx
AA
BB
xx
xx

Examplary structure of a PGN message

To design the structure oft the shown message the mapping and static mask register have following values:

register	value
Static mask 1 (0x0810)	0xBBAAFFFF
Static mask 2 (0x0811)	0xFFFFFFFF
Mapping entries 1 (0x0820)	0x0000002A
Mapping entries 2 (0x0821)	0x0000000B

10. REMARKS

The behaviour described in this document corresponds to the internal standard of ADZ NAGANO GmbH.

It is possible to alter the behaviour according to the customer's needs, so that the pressure transducer may be installed easily to the target application.

11. BIBLIOGRAPHY

[1] Stefan-Xp, "Wikimedia Commons (CC BY-SA 3.0)" 24 Nov. 2016.
[Online] Available: https://commons.wikimedia.org/w/index.php?curid=3607670.