Open SAE J1939

Foreword

SAE J1939 is a protocol for shaping the CAN-bus message in a specific way that suits industrial vehicles such as tractors, machinery, trucks and more.

SAE J1939 is a very easy protocol to use, but there is a lack of information about SAE J1939, due to the cost of the protocol document, available how to shape a CAN-bus message according to SAE J1939 protocol standard. So therefore I’m writing a SAE J1939 protocol available for free to use on any embedded systems such as STM32, Arduino, AVR, PIC etc.

To learn to build on this project, you need first to understand SAE J1939. I have written this project in C language because C is an industry standard. The C language dialect I have chosen is C99 and I don’t use dynamical memory allocation in this library. So it will work with MISRA C standard.

With this library, you can communicate with valves, engines, actuators, hardware and all other things that are suitable for heavy industrial mobile applications. I have build up a basic structure of the project and I hope that other users will send pull request of their C code for extra functionality to SAE J1939 standard because SAE J1939 is a huge standard.

- Daniel Mårtensson, Sweden, 2021-07-14

Quick introduction to SAE J1939 ID and DATA

First you need to understand the data frame of SAE J1939. The data frame of SAE J1939 is basically one ID and one DATA. ID and DATA is the information we are going to send through the CAN-bus network.

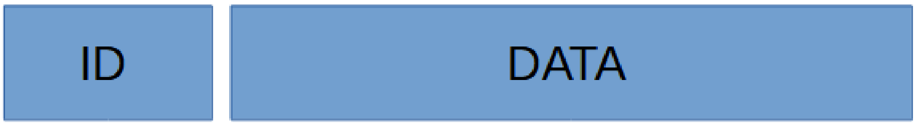


Figure 1: SAE J1939 Data frame

ID contains 4 hex values and DATA contains maximum 8 hex values e.g 8 bytes values.
An ID can be shaped as this 0x18FE8022 where 80 is the destination address, called DA and 22 is the source address, called SA. So you more likely going to see ID numbers as this 0x18FE’DA’’SA’

The DA address and SA address is the addresses of the ECU – Electronic Control Unit, e.g microcontroller/PCB board. You self decide what address your ECU should have. For example. If you have two ECU that are going to communicate to each other. Then you have for example ECU1 address as 0xA1 and ECU2 as 0xA5. If ECU1 what to send a message to ECU2, then the destination address DA would be 0xA5 and the source address SA would be 0xA1. If ECU2 want to send a message to ECU1, then DA would be 0xA1 and source address SA would be 0xA5.

So keep that in mind that when I’m talking about SA and DA, it’s only about source address and destination address of the ECU. When it comes to the rest of the ID values 0x18FE, they are just there to describe the type of message. Each message have a unique ID and therefore the ID will be shaped differently. Sometimes, DA in the ID message is replaced by a unique hex value.

The length of DA and SA are from 0x0 to 0xFD because 0xFE is error address and 0xFF is broadcast address.

DATA is just 8 bytes of data. DATA holds information about anything the user want to send to other ECU. Sometimes DATA can be 3 bytes depending on if we are sending a unique message.

Functionality overview of Open SAE J1939

Open SAE J1939 is just basic C-code, arrays, function, structures and bitwise operations. Nothing more. I have shaped this project according to this map below. This map describe basic overview of the project. Some functions may have been added after this document was made, but the structure is the same. Those functions are inside the green box and they are called extra functionality of the project.

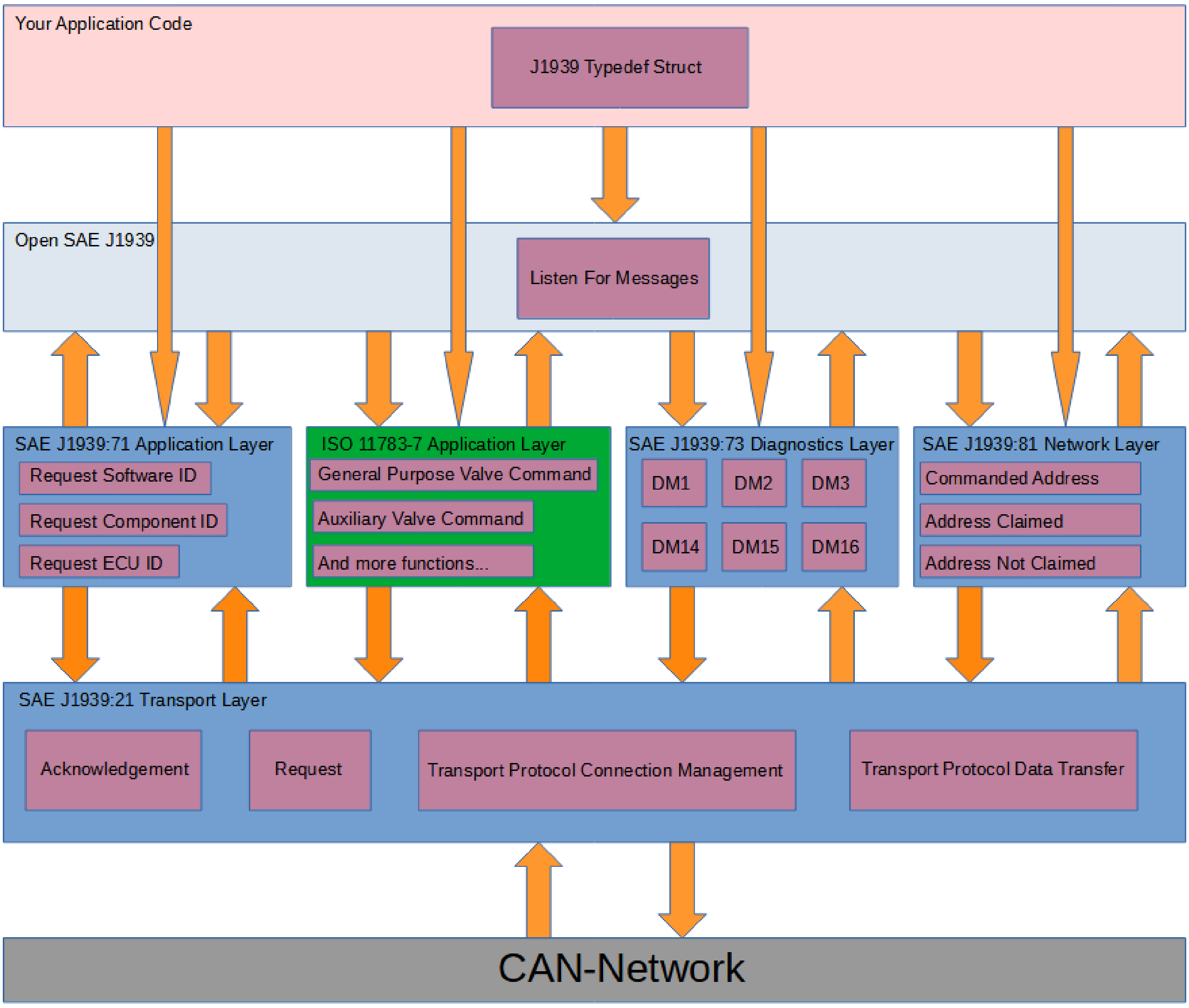


Figure 2: Structure of this project

We begin from the top. The J1939 Typedef Struct. That structure is just a basic C code typedef struct that holds information about all ECU who are connected to the CAN-network. The struct can be found at the Open SAE J1939 folder.

The idea behind the struct:

- The struct holds NAME + source address of all ECU
- The user are going to create a C pointer of the struct and pass it to the SAE J1939 functions.
- The struct hold temporary information about other ECU

Open SAE J1939

Listen For Messages

This function is called all the time by the user. Most likely inside a while loop. This function reading ID and DATA directly and split ID into 4 different ID for determine what kind of ID it is and what to do next. It split up ID, check what ID it is and then check what function it want to call and then pass that DATA + source address SA + pointer of the struct to that function as argument. That's it. Listen For Messages is just a function who select which function we want to call, depending on how the ID looks like.

SAE J1939:71 Application Layer

Request Software ID

Every ECU has a software ID for example a number or a text that describe for example BIOS software or software version. Call this function if you want to request the software ID from other ECU. The ECU will then response back to you with information.

Request Component ID

Every ECU has a component it control. It can be for example a motor, valve or other thing. Call this function if you want the ECU to send information about the component that ECU controls.

Request ECU ID

Same as above. Call this function if you want to know the ID information about the ECU you are sending to.

ISO 11783-7 Application Layer

Here I have selected the application layer of agriculture, tractors and machinery ISO standard. That application layer contains these functions:

- General Purpose Valve command – Sending control messages to ECU for controlling valves
- Auxiliary Valve Command – Sending control messages to ECU for controlling valves
- General Purpose Valve Estimated Flow – Get the computed flow from a valve
- Auxiliary Valve Estimated Flow – Get the computed flow from a valve
- Auxiliary Valve Estimated Position – Get the computed position of the valve main spool

SAE J1939:73 Diagnostics Layer

Diagnostics layer contains messages about errors, but it also can contains functionality such as memory access of the EEPROM (built in “hard drive”) or command for deleting error messages.

- DM1 – Contains error messages such as electrical fault, location etc
- DM2 – Contains previous active DM1 messages
- DM3 – Deleting DM2 messages
- DM14 – Memory request of bytes, address etc from a ECU
- DM15 – Memory response from a DM14 request from that ECU
- DM16 – Binary data transfer (if DM15 repose was OK)

SAE J1939:81 Network Layer

At the startup of an ECU, the ECU sending Address Claimed, which means it giving out the information about something called NAME. NAME is basic information about the ECU + SA.

- Address Claimed – Send out NAME + ID that holds SA.
- Address Not Claimed – Send out NAME + ID that holds Address Not Claimed.
- Commanded Address – Send this to an ECU and you can change its NAME and SA.

SAE J1939:21 Transport Layer

- Acknowledgement – Send a OK, BUSY, WAIT etc. as a response on a request.
- Request – Asking information or functionality about an ECU
- Transport Protocol Connection Management – Send how much data in packages you want to send to the receiver.
- Transport Protocol Data Transfer – Send the data in packages to the receiver.

SAE J1939 Protocol

Request

Request are used when we want to ask an ECU of “something”. That “something” is a PGN number. Enter a PGN number, send the message and get a response from that ECU directly. All PGN numbers can be found in SAE_J1939_Enum_PGN.h

Message ID	18 EA "DA" "SA"
PGN	0x00EA00 (59904)
Peer to Peer/Broadcast	Peer to Peer or Broadcast with DA = 0xFF (Used for Address Claimed)
Message length	3 bytes
Multipacket	No
Byte 1	PGN LSB

Byte 2	PGN
Byte 3	PGN MSB

Field	Max value	Min value	Explanation	Enum
PGN	0x0	0xFFFFFF	Request command	SAE_J1939_Enum_PGN.h

In case you did not understand what LSB and MSB is. LSB stands for least signifiant bit and MSB stands for most signifiant bit. LSB means the lowest number and MSB means the highest number. If we want to send the PGN number 0x18EEA1, then the LSB is 0xA1 because that’s the lowest number and MSB is 0x18 because that’s the highest number. The same reason when you write out 1024 where you begin with highest number 1(1000), then 0(000), then 2(20) and last 4. 1000 + 000 + 20 + 4 = 1024. So keep that in mind that MSB is the highest number and LSB is the lowest number.

Acknowledgement

Acknowledgement are used when we want to response back to an ECU e.g completed, busy, not available etc.

Message ID	18 E8 "DA" "SA"
PGN	0x00E800 (59392)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No
Byte 1	Control byte
Byte 2	Group function value
Byte 3	0xFF (Reserved)
Byte 4	0xFF (Reserved)
Byte 5	Address
Byte 6	PGN of requested info LSB
Byte 7	PGN of requested info
Byte 8	PGN of requested info MSB

Field	Max value	Min value	Explanation	Enum
Control byte	0x0	0xFF	Describes the status of the requested PGN	SAE_J1939_Enum_Control_Byte.h

Group function value	0x0	0xFF	The function code that specify the cause of the control byte	SAE_J1939_Enum_Group_Function_Value.h
Address	0x0	0xFF	The source address of the ECU from where the acknowledgement came from	
PGN of requested info	0x0	0xFFFFFF	The same PGN number in the request	SAE_J1939_Enum_PGN.h

Transport Protocol Connection Management

This describe for the receiver ECU how many packages and total size of data we are going to send via Transport Protocol Data Transfer.

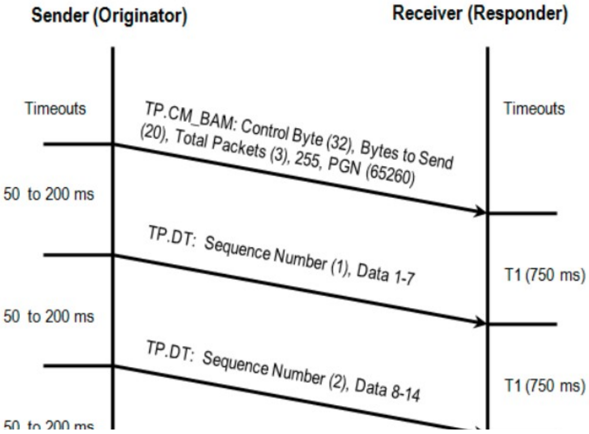
Message ID	1C EC "DA" "SA"
PGN	0x00EC00 (60416)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No
Byte 1	Control byte
Byte 2	Total message size LSB
Byte 3	Total message size MSB
Byte 4	Number of packages
Byte 5	0xFF (Reserved)
Byte 6	PGN of the packeted message LSB
Byte 7	PGN of the packeted message
Byte 8	PGN of the packeted message MSB

Field	Max value	Min value	Explanation	Enum
Control byte	0x0	0xFF	Describes how the data transfer should become	SAE_J1939_Enum_Control_Byte.h
Total message	0x9	0x6F9	Total bytes we are going to transfer	

size				
Number of packages	0x2	0xE0	How many packages are we going to transfer	
PGN of the packeted message	0x0	0xFFFFFF	The type of information we are sending	SAE_J1939_Enum_PGN.h

Here is two way to send a multi pack messages:

- BAM – Just send the message to the ECU
- RTS/CTS – Wait for a response (With an acknowledgement response at the end)


Sender (Originator)
Receiver (Responder)
Timeouts
TP.CM_BAM: Control Byte (32), Bytes to Send (20), Total Packets (3), 255, PGN (65260)
Timeouts
50 to 200 ms
TP.DT: Sequence Number (1), Data 1-7
T1 (750 ms)
50 to 200 ms
TP.DT: Sequence Number (2), Data 8-14
T1 (750 ms)
50 to 200 ms

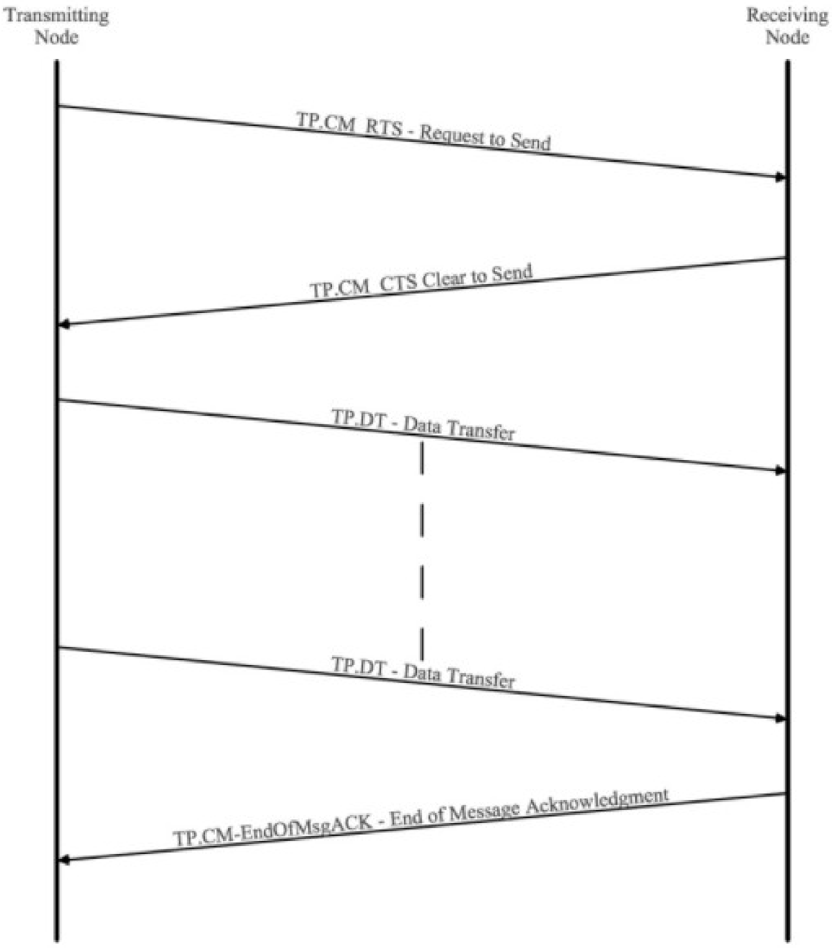


Figure 4: Request To Send(RTS), Clear To Send(CTS) and EndOfMsgACK

Transport Protocol Data Transfer

This sending packages to the receiver ECU. Transport Protocol Connection Management must be sent first. Else the receiver won’t know how many packages and total data and PGN number we the transmitter is sending.

Message ID	1C EB "DA" "SA"
PGN	0x00EB00 (60160)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No
Byte 1	Sequence number
Byte 2	Data 1
Byte 3	Data 2
Byte 4	Data 3
Byte 5	Data 4
Byte 6	Data 5
Byte 7	Data 6
Byte 8	Data 7

Field	Max value	Min value	Explanation	Enum
Sequence number	0x0	0xE0	This is the order/index of the data transfer packages	
Data 1 to Data 7	0x0	0xFF	Your data bytes	

Address Claimed

This describes the NAME about the ECU, and also the other ECU get to know its SA.

Message ID	18 EE FF "SA"
PGN	0x00EE00 (60928)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No
Repetition rate	Sent at power on, after a “Request” for “Address Claim” and after a

	“Commanded Address”
Byte 1	Identity number LSB
Byte 2	Identity number
Byte 3	Manufacturer code LSB [8..6], Identity number MSB [5..1]
Byte 4	Manufacturer code MSB
Byte 5	Function instance [8..4], ECU Instance [3..1]
Byte 6	Function
Byte 7	Vehicle system [8..2], 0x1 (Reserved) [1]
Byte 8	Arbitrary address capable [8], Industry group [7..5], Vehicle system instance [4..1]

Notice that [X..Y] describes the bit index in the byte. It more likely looks like this.

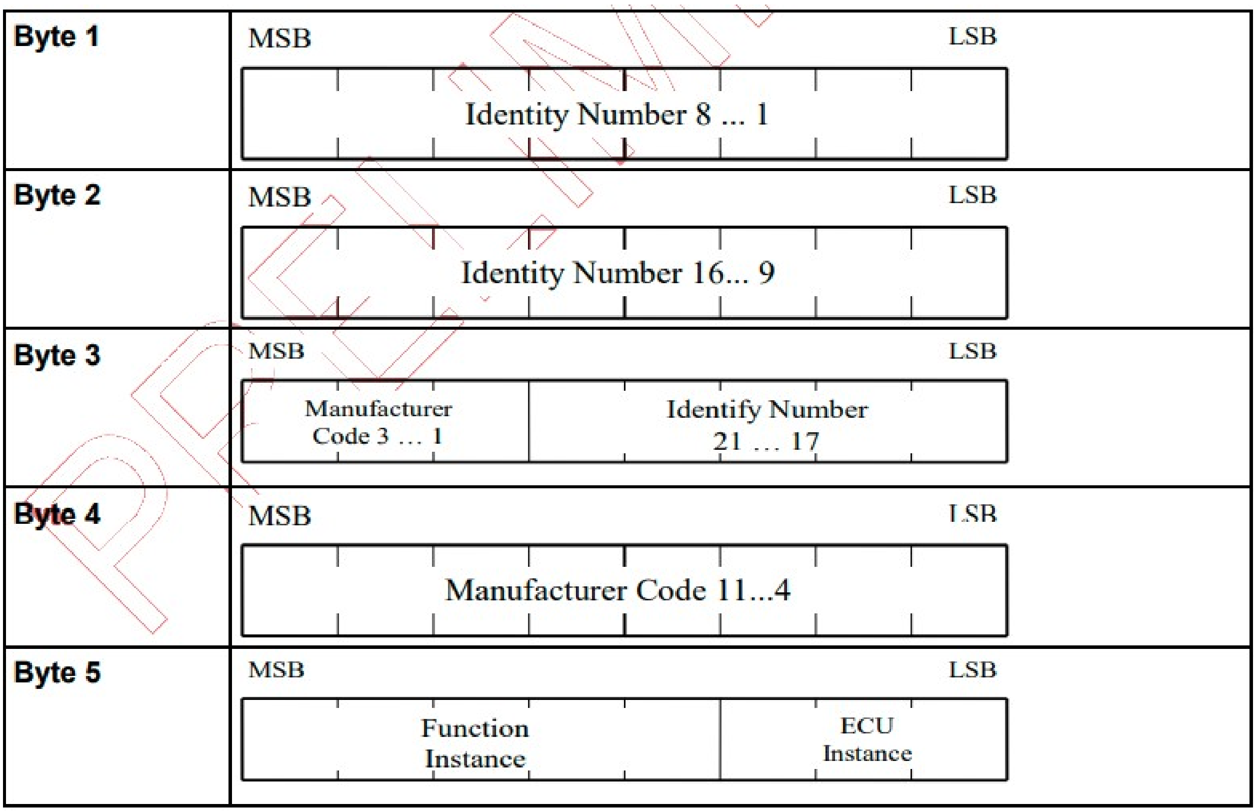


Figure 5: How to interpret

Field	Max value	Min value	Explanation	Enum
Identity number	0x0	0x1FFFFF	ECU identity number	
Manufacturer code	0x0	0x7FF	ECU manufacturer code	
Function	0x0	0x1F	ECU function area	

instance				
ECU instance	0x0	0x7	ECU function area	
Function	0x0	0xFF	ECU functionality	SAE_J1939_Enum_NAME.h
Vehicle system	0x0	0x7F	Where in the vehicle system the ECU is located	
Arbitrary address capable	0x0	0x1	If the ECU have right to change address if the address conflicts with another ECU	SAE_J1939_Enum_NAME.h
Industry group	0x0	0x7	Where in the industry the ECU is located	SAE_J1939_Enum_NAME.h
Vehicle system instance	0x0	0xF	The vehicle system code	

Address not claimed

When an ECU cannot claim their address, the other ECU will receive this message. The user don’t know which ECU cannot claim their address. The user can only see the amount of how many ECU could not claim their address. See the uint8_t variable number_of_cannot_claim_address inside the struct J1939.

Message ID	18 EE FF FE
PGN	0x00EE00 (60928)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No
Repetition rate	Sent if ECU address conflicts with other addresses
Byte 1	Identity number LSB
Byte 2	Identity number
Byte 3	Manufacturer code LSB [8..6], Identity number MSB [5..1]
Byte 4	Manufacturer code MSB
Byte 5	Function instance [8..4], ECU Instance [3..1]
Byte 6	Function
Byte 7	Vehicle system [8..2], 0x1 (Reserved) [1]
Byte 8	Arbitrary address capable [8], Industry group [7..5], Vehicle system

	instance [4..1]

Commanded Address

This command is used when the user want to change the NAME + SA at an ECU. Notice that this PGN don’t have an ID because Commanded Address DATA is 9 bytes and therefore using transport protocol data transfer.

PGN	0x00FED8 (65240)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	Yes. Use Transport Protocol Connection Management → Transport Protocol Data Transfer
Byte 1	Identity number LSB
Byte 2	Identity number
Byte 3	Manufacturer code LSB [8..6], Identity number MSB [5..1]
Byte 4	Manufacturer code MSB
Byte 5	Function instance [8..4], ECU Instance [3..1]
Byte 6	Function
Byte 7	Vehicle system [8..2], 0x1 (Reserved) [1]
Byte 8	Arbitrary address capable [8], Industry group [7..5], Vehicle system instance [4..1]
Byte 9	New ECU address

Field	Max value	Min value	Explanation	Enum
New ECU address	0x0	0xFD	New address of the ECU	

DM1

Diagnostics of error and location of an ECU. Notice that this message ID don’t have a destination address because it’s a broadcast message to all ECU.

Message ID	18 FE CA "SA"
PGN	0x00FECA (65226)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No: Normal transfer Yes: Use Transport Protocol Connection Management → Transport

	Protocol Data Transfer
Repetition rate	On request from other ECU
Byte 1	SAE Lamp status malfunction indicator [8..7], SAE Lamp status red stop [6..5], SAE Lamp status amber warning [4..3], SAE lamp status protect lamp [2..1]
Byte 2	SAE Flash lamp malfunction indicator [8..7], SAE Flash lamp red stop [6..5], SAE Flash lamp amber warning [4..3], SAE Flash lamp protect lamp [2..1]
Byte 3	SPN LSB
Byte 4	SPN
Byte 5	SPN MSB [8..6], FMI [5..1]
Byte 6	SPN conversion method [8], Occurence count[7..1]
Byte 7	0xFF (Reserved)
Byte 8	0xFF (Reserved)

Field	Max value	Min value	Explanation	Enum
SAE Lamp status malfunction indicator	0x0	0x1	SAE lamp indicates fault	
SAE Lamp status red stop	0x0	0x1	SAE lamp becomes red	
SAE Lamp status amber warning	0x0	0x1	SAE lamp warns with amber light	
SAE lamp status protect lamp	0x0	0x1	SAE lamp lights up the protection light	
SAE Flash lamp malfunction indicator	0x0	0x1	SAE flash lamp indicates fault	
SAE Flash Lamp red stop	0x0	0x1	SAE flash lamp becomes red	
SAE Flash	0x0	0x1	SAE flash lamp warns	

Lamp amber warning			with amber light	
SAE Flash lamp protect lamp	0x0	0x1	SAE flash lamp lights up the protection light	
SPN	0x0	0x7FFFFF	Suspect Parameter Number – Location of the fault	SAE_J1939_Enum_DM1.h
FMI	0x0	0x1F	Failure Mode Identifier – What cause the fault	SAE_J1939_Enum_DM1.h
SPN Conversion method	0x0	0x1	1 = Diagnostics Trouble Code are aligned using a newer conversion method. 0 = One of the three Diagnostics Trouble Code conversion methods is used and ECU manufacture shall know which of the three methods is used	
Occurence count	0x0	0x7E	Count how often the DM1 error message becomes active	

DM2

DM2 is the previous active DM1 messages.

Message ID	18 FE CB "SA"
PGN	0x00FECB (65227)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No: Normal transfer Yes: Use Transport Protocol Connection Management → Transport Protocol Data Transfer
Repetition rate	On request from other ECU
Byte 1	SAE Lamp status malfunction indicator [8..7], SAE Lamp status red stop [6..5], SAE Lamp status amber warning [4..3], SAE lamp status protect lamp [2..1]
Byte 2	SAE Flash lamp malfunction indicator [8..7], SAE Flash lamp red stop [6..5], SAE Flash lamp amber warning [4..3],

	SAE Flash lamp protect lamp [2..1]
Byte 3	SPN LSB
Byte 4	SPN
Byte 5	SPN MSB [8..6], FMI [5..1]
Byte 6	SPN conversion method [8], Occurence count[7..1]
Byte 7	0xFF (Reserved)
Byte 8	0xFF (Reserved)

DM3

DM3 is a request to clear DM2 messages. Use the request function above. After this is done, DM2 will be sent to all ECU via broadcast.

PGN	0x00FECC (65228)

DM11 – Not supported

DM11 is a request for clearing DM1 messages. The reason why I don't have implement it, is because in my opinion, it's just an administrative burden that other ECU need to clear the DM1 error codes. I want the ECU it self to clear their own DM1 codes. The same way it activate the error codes by it self.

Software identification

Send out the software ID of the ECU. Notice that DA here in the message ID is not destination address. It's the hex value 0xDA.

Message ID	18 FE DA "SA"
PGN	0x00FEDA (65242)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No: Normal transfer Yes: Use Transport Protocol Connection Management → Transport Protocol Data Transfer
Repetition rate	On request from other ECU
Byte 1	Number of fields
Byte 2-N	Identification

Field	Max value	Min value	Explanation	Enum

Number of fields	0x0	0x1E	How many fields we are going to transfer	
Identification 1 to N	0x0	0xFF	Each identification is one byte e.g ASCII	

ECU identification

Send out ECU identification to the ECU.

Message ID	18 FD C5 "SA"
PGN	0x00FDC5 (64965)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No: Normal transfer Yes: Use Transport Protocol Connection Management → Transport Protocol Data Transfer
Repetition rate	On request from other ECU
Byte 1	ECU part number
Byte 2	ECU serial number
Byte 3	ECU location
Byte 4	ECU type
Byte 5	0xFF (Reserved)
Byte 6	0xFF (Reserved)
Byte 7	0xFF (Reserved)
Byte 8	0xFF (Reserved)

Notice that there are a field called length_of_each_field in ECU_identification inside the J1939 struct. If length_of_each_field = 1, then this message is going to be transfer in a normal way, else multipacket.

Field	Max value	Min value	Explanation	Enum
ECU part number	0x0	0xFF	The part number in ASCII format	
ECU serial number	0x0	0xFF	The serial number in ASCII format	
ECU location	0x0	0xFF	The location in ASCII format	
ECU type	0x0	0xFF	The type in ASCII format	

Component identification

Send out component identification to the ECU.

Message ID	18 FE EB "SA"
PGN	0x00FEEB (65259)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No: Normal transfer Yes: Use Transport Protocol Connection Management → Transport Protocol Data Transfer
Repetition rate	On request from other ECU
Byte 1	Component product date
Byte 2	Component model name
Byte 3	Component model number
Byte 4	Component unit name
Byte 5	0xFF (Reserved)
Byte 6	0xFF (Reserved)
Byte 7	0xFF (Reserved)
Byte 8	0xFF (Reserved)

Notice that there are a field called length_of_each_field in Component_identification inside the J1939 struct. If length_of_each_field = 1, then this message is going to be transfer in a normal way, else multipacket.

Field	Max value	Min value	Explanation	Enum
Component product date	0x0	0xFF	The product date in ASCII format	
Component model name	0x0	0xFF	The model name in ASCII format	
Component model number	0x0	0xFF	The model number in ASCII format	
Component unit name	0x0	0xFF	The unit name in ASCII format	

DM14 & DM15 and DM16

DM14 is a memory request. DM15 is a memory response for the DM14 memory request and if DM15 response was proceeded, then DM16 data transfer is called next.

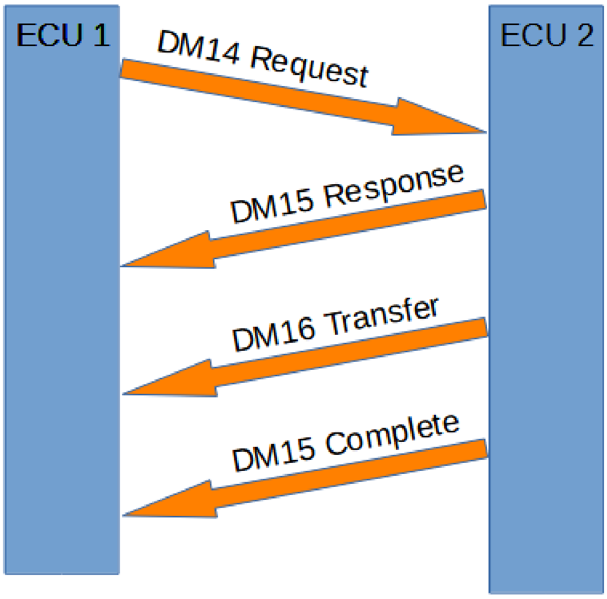


Figure 5: DM14, DM15 and DM16

DM14 memory request:

Message ID	18 D9 "DA" "SA"
PGN	0x00D900 (55552)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No
Byte 1	Number of requested bytes LSB
Byte 2	Number of requested bytes MSB [8..6], Pointer type [7], Command [6..3], 0x1 (Reserved) [1]
Byte 3	Pointer LSB
Byte 4	Pointer
Byte 5	Pointer MSB
Byte 6	Pointer extension
Byte 7	Key LSB
Byte 8	Key MSB

Field	Max value	Min value	Explanation	Enum
Number of requested bytes	0x0	0xFF	The question of how many bytes the ECU want to have	
Pointer type	0x0	0x1	0 if Pointer and Pointer extension are together one addresses. 1 if pointer is an address and Pointer extension is a way to describe a function code	SAE_J1939_Enum_DM14_DM15.h
Command	0x0	0x7	The command what to do	SAE_J1939_Enum_DM14_DM15.h
Pointer	0x0	0xFFFFFF	The memory address	
Pointer extension	0x0	0xFF	Extra memory address, or it can be a function code	SAE_J1939_Enum_DM14_DM15.h
Key	0x0	0xFFFF	Password or no password at all	SAE_J1939_Enum_DM14_DM15.h

DM15 memory response of the DM14 memory request:

Message ID	18 D8 "DA" "SA"
PGN	0x00D800 (55296)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No
Repetition rate	On request from other ECU
Byte 1	Number of allowed bytes LSB
Byte 2	Number of allowed bytes MSB [8..6], 0x1 (Reserved) [7], Status [6..3], 0x1 (Reserved) [1]
Byte 3	EDC parameter LSB
Byte 4	EDC parameter
Byte 5	EDC parameter MSB
Byte 6	EDCP extension
Byte 7	Seed LSB
Byte 8	Seed MSB

Field	Max value	Min value	Explanation	Enum
Number of allowed bytes	0x0	0xFF	The question of how many bytes the ECU is going to have	
Status	0x0	0x7	The response status of the request	SAE_J1939_Enum_DM14_DM15.h
EDC parameter	0x0	0xFFFFFF	This is an explanation of the status response	SAE_J1939_Enum_DM14_DM15.h
EDCP extension	0x0	0xFF	This is an explanation of the EDC parameter	SAE_J1939_Enum_DM14_DM15.h
Seed	0x0	0xFFFF	Status of the key request	SAE_J1939_Enum_DM14_DM15.h

DM16 is binary data transfer:

Message ID	18 D7 "DA" "SA"
PGN	0x00D700 (55040)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No: Normal transfer Yes: Use Transport Protocol Connection Management → Transport Protocol Data Transfer
Byte 1	Number of occurrences
Byte 2-256	Raw binary data 1-255

Field	Max value	Min value	Explanation	Enum
Number of occurrences	0x0	0xFF	How many raw binary data we have	
Raw binary data 1-255	0x0	0xFF	Data to transfer	

Auxiliary Valve Command

Auxiliary valve command is a command from an ECU to broadcast to all other ECU. All it does it to tell other ECU which valve they are going to use and how they are going to use it. Maximum 16 valve numbers can be used, from 0 to 15 according to ISO 11783-7 standard.

Message ID	0C FE (30+valve number) "SA"
PGN	0x00FE(30+valve number) (65072+valve number)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No

Byte 1	Standard flow
Byte 2	Fail safe mode [8..7], 0x3 (Reserved) [6..5], Valve state [4..1]
Byte 3	0xFF (Reserved)
Byte 4	0xFF (Reserved)
Byte 5	0xFF (Reserved)
Byte 6	0xFF (Reserved)
Byte 7	0xFF (Reserved)
Byte 8	0xFF (Reserved)

Field	Max value	Min value	Explanation	Enum
Standard flow	0x0	0xFF	In practice, this is the position of the valve main spool	
Fail safe mode	0x0	0x1	1 = Spool go to neutral 0 = No fail safe mode	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Valve state	0x0	0xF	What way the valve should act	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h

General Purpose Valve Command

The difference between Auxiliary Valve Command and General Purpose Valve Command is that Auxiliary Valve Command can hold 16 valves meanwhile General Purpose Valve Command can hold 1 valve.

Message ID	0C C4 "DA" "SA"
PGN	0x00C400 (50176)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No
Byte 1	Standard flow
Byte 2	Fail safe mode [8..7], 0x3 (Reserved) [6..5], Valve state [4..1]
Byte 3	Extended flow LSB
Byte 4	Extended flow MSB
Byte 5	0xFF (Reserved)
Byte 6	0xFF (Reserved)
Byte 7	0xFF (Reserved)

Byte 8	0xFF (Reserved)

Field	Max value	Min value	Explanation	Enum
Standard flow	0x0	0xFF	In practice, this is the position of the valve main spool	
Fail safe mode	0x0	0x1	1 = Spool go to neutral 0 = No fail safe mode	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Valve state	0x0	0xF	What way the valve should act	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Extended flow	0x0	0xFFFF	In practice, this is the position of the valve main spool in higher precision	

Auxiliary Valve Estimated Flow

Broadcast estimated flow from an auxiliary valve. Total 16 valves can be used, from 0 to 15.

Message ID	0C FE (10+valve number) "SA"
PGN	0x00FE(10+valve number) (65040+valve number)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No
Repetition rate	On request from other ECU
Byte 1	Extend estimated flow standard
Byte 2	Retract estimated flow standard
Byte 3	Fail safe mode [8..7], 0x3 (Reserved) [6..5], Valve state [4..1]
Byte 4	Limit [8..6], 0x1F (Reserved)
Byte 5	0xFF (Reserved)
Byte 6	0xFF (Reserved)
Byte 7	0xFF (Reserved)
Byte 8	0xFF (Reserved)

Field	Max value	Min value	Explanation	Enum
Extend estimated flow	0x0	0xFF	In practice, this is the flow of the valve main spool	

standard				
Retract estimated flow standard	0x0	0xFF	In practice, this is the negative flow of the valve main spool	
Fail safe mode	0x0	0x1	1 = Spool go to neutral 0 = No fail safe mode	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Valve state	0x0	0xF	What way the valve should act	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Limit	0x0	0x7	Limit code of the valve	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h

General Purpose Valve Estimated Flow

Broadcast estimated flow from a general purpose valve.

Message ID	0C C6 "DA" "SA"
PGN	0x00C600 (50688)
Peer to Peer/Broadcast	Peer to Peer
Message length	8 bytes
Multipacket	No
Repetition rate	On request from other ECU
Byte 1	Extend estimated flow standard
Byte 2	Retract estimated flow standard
Byte 3	Fail safe mode [8..7], 0x3 (Reserved) [6..5], Valve state [4..1]
Byte 4	Limit [8..6], 0x1F (Reserved)
Byte 5	Extend estimated flow extended LSB
Byte 6	Extend estimated flow extended MSB
Byte 7	Retract estimated flow extended LSB
Byte 8	Retract estimated flow extended MSB

Field	Max value	Min value	Explanation	Enum
Extend estimated flow standard	0x0	0xFF	In practice, this is the flow of the valve main spool	
Retract estimated flow standard	0x0	0xFF	In practice, this is the negative flow of the valve main spool	

Fail safe mode	0x0	0x1	1 = Spool go to neutral 0 = No fail safe mode	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Valve state	0x0	0xF	What way the valve should act	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Limit	0x0	0x7	Limit code of the valve	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h
Extend estimated flow extended	0x0	0xFFFF	In practice, this is the high precision flow of the valve main spool	
Retract estimated flow extended	0x0	0xFFFF	In practice, this is the negative high precision flow of the valve main spool	

Auxiliary Valve Measured Position

Broadcast measured position from an auxiliary valve. Total 16 valves can be used, from 0 to 15.

Message ID	0C FF (20+valve number) "SA"
PGN	0x00FF(20+valve number) (65312+valve number)
Peer to Peer/Broadcast	Broadcast
Message length	8 bytes
Multipacket	No
Repetition rate	On request from other ECU
Byte 1	Measured position procent LSB
Byte 2	Measured position procent MSB
Byte 3	Valve state
Byte 4	Measured position micrometer LSB
Byte 5	Measured position micrometer MSB
Byte 6	0xFF (Reserved)
Byte 7	0xFF (Reserved)
Byte 8	0xFF (Reserved)

Field	Max value	Min value	Explanation	Enum
Measured position procent	0x0	0xFFFF	In practice, this is the position in procent of main spool	
Valve state	0x0	0xF	What way the valve should act	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h

Measured position micrometer	0x0	0xFFFF	In practice, this is the position in micrometer of main spool	ISO_11783_Enum_Auxiliary_And_General_Purpose_Valves.h