SAE International™

SURFACE VEHICLE INFORMATION REPORT

SAE J2819 FEB2008

Issued 2008-02

TP2.0 Vehicle Diagnostic Protocol

RATIONALE

Some Volkswagen of America and Audi of America vehicles are equipped with ECU(s), in which a TP2.0 proprietary diagnostic communication protocol is implemented. This document is needed to specify the requirements necessary to implement the TP2.0 communication protocol in an SAE J2534 interface.

TABLE OF CONTENTS

TO PLACE A DOCUMENT ORDER: **Tel: 877-606-7323 (inside USA and Canada)**
Tel: 724-776-4970 (outside USA)
Fax: 724-776-0790
Email: CustomerService@sae.org
SAE WEB ADDRESS: http://www.sae.org

1. SCOPE

This Technical Information Report defines the diagnostic communication protocol TP2.0. This document should be used in conjunction with SAE J2534-2 in order to fully implement the communication protocol in an SAE J2534 interface.

Some Volkswagen of America and Audi of America vehicles are equipped with ECU(s), in which a TP2.0 proprietary diagnostic communication protocol is implemented. The purpose of this document is to specify the requirements necessary to implement the communication protocol in an SAE J2534 interface.

2. REFERENCES

2.1 Applicable Publications

The following publications form a part of this specification to the extent specified herein. Unless otherwise specified, the latest issue of SAE publications shall apply.

2.1.1 SAE Publications

Available from SAE International, 400 Commonwealth Drive, Warrendale, PA 15096-0001, Tel: 877-606-7323 (inside USA and Canada) or 724-776-4970 (outside USA), www.sae.org.

SAE J2534 Recommended Practice for Pass-Thru Vehicle Programming

SAE J2534-1 Recommended Practice for Pass-Thru Vehicle Programming

3. TERMS AND ACRONYMS

Ack	Acknowledge message
AR	Acknowledge request
BR	Break message
BS	Block size
CA	Connection acknowledge message
CAN Telegram	A CAN message with 11 address bits and up to 8 data bytes used to request, respond or establish a connection.
CS	Connection set-up message
CT	Connection test message
DC	Disconnect message
DT	Data message
ECU	Electronic Control Module
EOM	End of message
ID or Identifier	Fixed CAN ID assigned to a module. Range 0x200 through 0x2EF
MNCT	Maximum value during connection test
MNT	Maximum value during communication
MNTB	Maximum value at Block repeat count
MNTC	Maximum value with connection structure
RBR	Break
RC_CT	Repeat counter connection test
RC_E	Repeat counter opening
RC_S	Repeat counter sending machine
RNR	Receiver not ready
RR	Receiver ready
RS	Receiver state
SID	Service identifier
SN	Block counter
T_CT	Timer for connection test
T_E	Time-out for channel structure
T_WAIT	Delay timer for RNR

T1, T3	Parameter ECU (CS / CA)
TP	Transport protocol
TPCI	Transport protocol control information byte
TPDU	Transport protocol data unit

“Re-triggered”:
Service requests that are Re-triggered are Broadcast Messages service requests that are sent continuously at an interval of T_BRT_INT until stopped by the user (application).

“Active”:
An active module is the module that is sending data to a passive module.

“Passive”:
A passive module is the module that is receiving data from an active module.

“Identifier”:
Each message in TP2.0 has an identifier which is the 11 bit Address of the CAN message. Each module on the CAN bus has been assigned fixed Identifier in the range of 0x200 through 0x2EF. Dynamic channels use dynamic assigned CAN addresses.

4. OVERVIEW

This document describes the transport protocol and the broadcast services. The TP2.0 is an exclusive connection between two CAN (11 bit IDs only) participants for the transmission of large amounts of data. The broadcast services are used for the 1:n communication in the vehicle.

The CAN-transport protocol includes an agreement for the dynamic assignment of bi- directional transport channels between control modules. It is an extension of the transport protocol that was standardized in the OSEK-communication V1.0.

The generalization of the OSEK- connection is necessary to make dynamic assignments of identifiers to transport connections and to make a discontinuation of a running data transmission and additional timings possible.

For the dynamic identifier a unique address was assigned to each control module for all vehicles and a firm question or answer address channel. By exchanging messages, the systems assign on these channels the transport channels that must then be used.

The major attributes of these transport protocols are:

- Control bytes for channel structure, connection structure, structure confirmation, connection control, data transfer and confirmation,

- Pure bi-directional channels,

- Confirmation of each telegram or major block of telegrams including error correction,

- Interruption of a running data transmission

5. CAN MESSAGE FORMAT

5.1 CAN-Telegram Overview

Each ECU in a vehicle has been assigned a TP-target address. The TP-target address range is from 0x00 to 0xEF. The Broadcast Addresses are TP-target addresses in the range of 0xF0 through 0xFF.

Each CAN-telegram has the following basic structure:

Identifier		1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
11 bits		Destination	Opcode	Parameter				

Identifier: Fixed address of the sending (active) ECU. Range between 0x200 through 0x2EF.

Destination: TP-target address of the recipient (lower 8 bits of the ECU's assigned Identifier)

Opcode:

Broadcast message

Request 0x23

Response 0x24

Dynamic channel structure message

Channel Set-up 0xC0

Channel Ack-Positive Reply 0xD0

Channel Ack-Negative Reply 0xD6: Application type not supported

Channel Ack-Negative Reply 0xD7: Application type temporarily not supported

Channel Ack-Negative Reply 0xD8: Temporarily no resources are free

Parameter: The Parameter data area is assigned according to the opcode.

5.1.1 Non-Broadcast Request Messages

Non-Broadcast Request messages are defined as messages with a Destination Byte less than 0xFx.

Identifier	1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
10 -0	7-0	7-0	7-0	7-0	7-0	7-0	7-0
11 bits	Destination < 0xFx	0x23	SID	Param1	Param2	Key	

SID: Desired Service Request

Param1 and Param2: Parameters for Service requested

KeyValue: Value = 0

The Destination module should respond with T_RSP.

Response:

Identifier	1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
10 –0	7-0	7-0	7-0	7-0	7-0	7-0	7-0
11 bits	Destination	0x24	SID	Param1	Param2	Param3	Param4

SID: Service Request

Param1 – Param4: Response Parameters for Service requested

5.1.2 Broadcast Messages

Broadcast messages are defined as messages with a Destination Byte of 0xFx.

Identifier	1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
10 –0	7-0	7-0	7-0	7-0	7-0	7-0	7-0
11 bits	Destination = 0xFx	0x23	SID	Param1	Param2	Key Value x	

SID: Desired Service Request
Param1 and Param2: Parameters for Service requested
Key Value x: Value that must change between Key1 and Key 2 on each transmission of the message. Values: Key Value 1 = 0x5555; Key Value 2 = 0xAAAA

For a single (not Re-triggered) Service Request, the Service Request message is sent 5 times, T_BR_INT milliseconds apart, with the key alternating between Key1 and Key2. The destination will consider the message received when it sees at least one message received with SIDwith Key Value 1 and one message received with SIDwith Key Value 2 within 100ms.

For Re-triggered service requests, one that is meant to set the addressed systems into a special mode, the service request is sent repeating 5 times every T_BR_INT ms as before but then has to be repeated at a larger sending grid (T_BRT_INT) as long as the condition must be retained. If a control module recognizes a time-out (2500ms) for the re-triggering, it leaves the special mode and returns to the regular operating mode. The key changes with every sending occurrence. Start-up is with key 0x5555.

Example showing the data portion of the messages:

t = 0ms

1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
Destination	0x23	SID	Param. 1	Param. 2	0x55	0x55

t = 20ms

1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
Destination	0x23	SID	Param. 1	Param. 2	0xAA	0xAA

t = 40ms

1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
Destination	0x23	SID	Param. 1	Param. 2	0x55	0x55

t = 60ms

1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
Destination	0x23	SID	Param. 1	Param. 2	0xAA	0xAA

t = 80ms

1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
Destination	0x23	SID	Param. 1	Param. 2	0x55	0x55

For Re-trigger service requests additional messages sent every T_BRT_INT ms.

t = 1080ms

1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
Destination	0x23	SID	Param. 1	Param. 2	0xAA	0xAA

t = 2080ms

1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte
Destination	0x23	SID	Param. 1	Param. 2	0x55	0x55

5.1.3 Dynamic Channel Structure Messages

Dynamic Channel messages are used to establish a data channel between 2 ECUs (or the tester and an ECU) for the purpose of sending large blocks of data. The Channel Set-up message is sent from the active ECU to a passive ECU to see if it can start a Dynamic channel in order to send the block of data to the ECU. The receiving passive ECU must send a positive or negative response to the message. The sending active ECU will wait for the response or time-out.

In the Channel Set-up message's data field, the requester sets the active RX-ID-A that it would like to use for this conversation. This is the CAN ID that the active ECU will use to receive messages. The RX-ID-A is an 11 bit CAN ID, the lower 8 bits are placed in RX-ID-A-Low (Byte 5 of the message) and the upper 3 bits are placed in RX-ID-A-High (bits 0 – 2 of byte 6).

The passive ECU will respond with a channel acknowledge message confirming the Active ECUs RX-ID-A and specifying the passive ECU's RX-ID-P. Note: the TX-ID-A must equal the RX-ID-P and the TX-ID-P must equal the RX-ID-A.

The coding of the Channel Set-up message is as follows:

Identifier	1. Byte	2. Byte	3. Byte	4. Byte		5. Byte	6. Byte		7. Byte
10-0	7-0	7-0	7-0	7-3	2-0	7-0	7-3	2-0	7-0
11 bits	Destination	0xC0	TX-ID-A-Low	Info-TX	TX-ID-A-High	RX-ID-A-Low	Info-RX	RX-ID-A-High	Application-Type

Definition of the bytes and bit fields

Byte	Bits	Name	Description
1	7 - 0	Destination	TP Target Address
2	7 - 0	Op Code 0xC0	Channel Setup
3	7 - 0	TX-ID-A-Low	Request:0; no specifications (because the active ECU does not know the passive ECUs ID yet)
4	7 - 3	Info-TX	Bit 3:reserved Bit 4:Request:1 = no specifications for TX-ID / Contents not to be evaluated Bit 5-7:reserved: 0
4	2 - 0	TX-ID-A-High	Request:0; no specifications
5	7 - 0	RX-ID-A-Low	Request:Bits 7-0 of the requested reception identifier from requesting module
6	7 - 3	Info-RX	Bit 3:reserved Bit 4:0 (RX-ID must be indicated) Bit 7-5:reserved: 0
6	2 - 0	RX-ID-A-High	Request:Bits 10-8 of the requested reception identifier
7	7 - 0	Application-Type	0x01 = SD Diagnostics, 0x10 = Infotainment communication, 0x20 = Application Protocol and 0x21 = WFS/WIV.

The coding of the Channel Ack-Positive Reply message is as follows:

Identifier	1. Byte	2. Byte	3. Byte	4. Byte		5. Byte	6. Byte		7. Byte
10-0	7-0	7-0	7-0	7-3	2-0	7-0	7-3	2-0	7-0
11 bits	Destination	0xD0	TX-ID-P-Low	Info-TX	TX-ID-P-High	RX-ID-P-Low	Info-RX	RX-ID-P-High	Application-Type

Definition of the bytes and bit fields

Byte	Bits	Name	Description
1	7 - 0	Destination	the lower 8 bits of the Identifier who sent the request
2	7 - 0	Op Code 0xD0	Channel Ack-Positive Reply
3	7 - 0	TX-ID-P-Low	Reply:Bits 7-0 must equal the RX-ID-A-Low bits.
4	7 - 3	Info-TX	Bit 3: reserved Bit 4: Reply:0 (TX-ID must be indicated) Bit 5-7:reserved: 0
4	2 - 0	TX-ID-P-High	Reply:Bits 10-8 must equal the RX-ID-A-High bits.
5	7 - 0	RX-ID-P-Low	Reply:Bits 7-0 of the established reception identifier of the passive ECU.
6	7 - 3	Info-RX	Bit 3:reserved Bit 4:0 (RX-ID must be indicated) Bit 7-5:reserved: 0
6	2 - 0	RX-ID-P-High	Reply:Bit 10-8 of the established reception identifier of the passive ECU
7	7 - 0	Application-Type	0x01 = SD Diagnostics, 0x10 = Infotainment communication, 0x20 = Application Protocol and 0x21 = WFS/WIV.

The responding ECU must be able to accept the RX-ID-A sent by the requester as the CAN ID the responder will use to send messages to the requester (TX-ID-P). The responding ECU returns the RX-ID-A as the TX-ID-P in the response or it must reject the channel request. The responder must fill in the RX-ID-P High and Low in the Channel Ack message. The RX-ID-P is the CAN ID that the passive ECU will use to receive messages. The responding passive ECU must be able to accept the application-type and return it in the response or reject the channel request.

After receiving a Channel Ack-Positive Reply, the sending and responding ECU use the Transport Protocol Data Unit Telegrams to send data and manage the channel.

The coding of the Channel Ack-Negative Reply message is as follows:

Identifier	1. Byte	2. Byte	3. Byte	4. Byte		5. Byte	6. Byte		7. Byte
10-0	7-0	7-0	7-0	7-3	2-0	7-0	7-3	2-0	7-0
11 bits	Destination	0xD6							

Destination: the lower 8 bits of the Identifier who sent the request.
The value of 0xD6 at 2 Byte indicates the passive module does not support the application type requested.

5.1.4 Static CAN-Telegram parameters

The following static CAN-Telegram parameters are defined to maintain the link.

TABLE 1 - STATIC CAN TELEGRAM PARAMETERS

Description	**Parameter Name**	**Value**
Broadcast		
Broadcast Destination address		0xF0 – 0xFF
Time between Broadcast messages	T_BR_INT	20 ms
Time between re-triggered Broadcast messages	T_BRT_INT	1000 ms
Request / Response		
Timeout waiting for a response from a request	T_RSP	500 ms
Dynamic Channel structure messages		
Maximum repeats of the connecting structure telegrams	MNTC	10
Time-out for connection structure waiting for response	T_E	100 ms

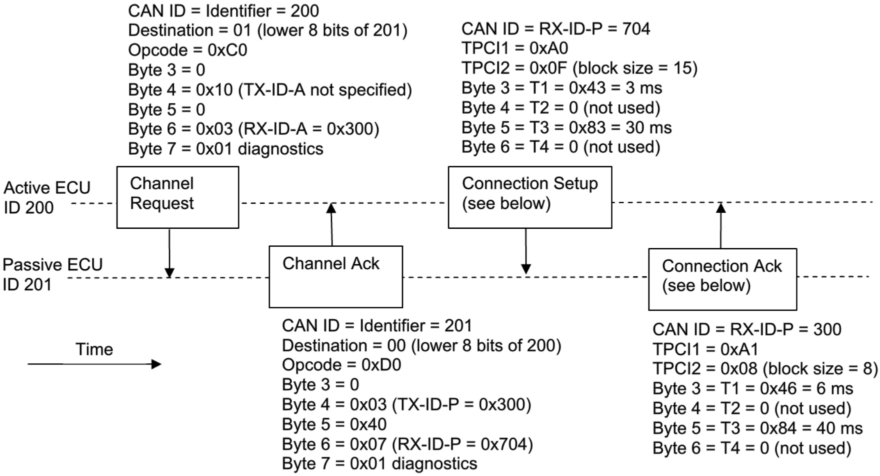
5.1.5 CAN-Telegram Error handling

TABLE 2 - CAN TELEGRAM ERROR HANDLING

Condition	**Action**
Time-out T_E exceeded waiting for Channel Ack	If retries < MNTC then send Connection Channel Set-up again
MNTC exceeded on waiting for Channel Ack	Give up with a Connection Failed

5.1.6 CAN-Telegram Establishing a Channel and Connection

Below is an example of how a channel and connection is established.



CAN ID = Identifier = 200
Destination = 01 (lower 8 bits of 201)
Opcode = 0xC0
Byte 3 = 0
Byte 4 = 0x10 (TX-ID-A not specified)
Byte 5 = 0
Byte 6 = 0x03 (RX-ID-A = 0x300)
Byte 7 = 0x01 diagnostics

CAN ID = RX-ID-P = 704
TPCI1 = 0xA0
TPCI2 = 0x0F (block size = 15)
Byte 3 = T1 = 0x43 = 3 ms
Byte 4 = T2 = 0 (not used)
Byte 5 = T3 = 0x83 = 30 ms
Byte 6 = T4 = 0 (not used)

CAN ID = Identifier = 201
Destination = 00 (lower 8 bits of 200)
Opcode = 0xD0
Byte 3 = 0
Byte 4 = 0x03 (TX-ID-P = 0x300)
Byte 5 = 0x40
Byte 6 = 0x07 (RX-ID-P = 0x704)
Byte 7 = 0x01 diagnostics

CAN ID = RX-ID-P = 300
TPCI1 = 0xA1
TPCI2 = 0x08 (block size = 8)
Byte 3 = T1 = 0x46 = 6 ms
Byte 4 = T2 = 0 (not used)
Byte 5 = T3 = 0x84 = 40 ms
Byte 6 = T4 = 0 (not used)

The active ECU sends a Channel Request message to the passive ECU. The passive ECU responds with a Channel Ack message. After this exchange, the both the active and passive ECUs have the channel CAN IDs which will be used by the new connection (RX-ID-A and RX-ID-P). A connection can now be established using the Connection Setup and Connection Ack messages.

5.2 Transport Protocol Data Unit Telegrams on an Established Channel

Once a Dynamic Channel has been established, use the Transport Protocol Data Unit (TPDU) messages to send the data and maintain the channel. 4 Transport Protocol Channels must be maintained simultaneously.

The coding of the Transport Protocol Data Unit (TPDU) Telegram message is as follows:

Identifier	1. Byte	2. Byte	3. Byte	4. Byte	5. Byte	6. Byte	7. Byte	8. Byte
10 -0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0
RX-ID-A or RX-ID-P	TPCI1	TPCI2 or Data or NA	Data or T1 or NA	Data or T2 or NA	Data or T3 or NA	Data or T4 or NA	Data or NA	Data or NA

The Active ECU will always use identifier RX-ID-P to send messages to the passive ECU. The passive ECU will always use identifier RX-ID-A to send messages to the active ECU.

The TPC1 byte defines 7 different TPDU telegram types:.

- Data: The Data telegram is used to send 7 or less bytes of data. A bit in the Data TPC1 tells the receiver if an Ack is expected after the Data telegram.

- Acknowledge: The Ack telegram is used to acknowledge the receipt of the Data telegrams. The SN field in the Ack message is used to request retransmission of data telegram. Normally, SN is equal to the last Data Telegram’s SN + 1. If the Ack’s SN is not equal to the Data telegram’s SN+1, the sender should re-send the data telegrams starting at the telegram who’s SN was specified in the Ack.

- Connection Set-up: The Connection Set-up telegram is used to start a transfer of data. The sending ECU who sends the Connection Set-up is considered the active ECU and is in control of the channel. The sending ECU sends it’s T1 and T3 values to the receiving ECU. The receiving ECU must always space all it’s messages back to the sending ECU by T3.

- Connection Acknowledge: The Connection Acknowledge telegram is sent in response to a Connection Set-up telegram or a Connection Test telegram. The receiving ECU sends it’s T1 and T3 values in this message. The Sending ECU must space it’s messages sent to the receiving ECU by T3.

- Connection Test: The Connection Test telegram is sent by the active ECU at the T_CTa interval. The Connection Test telegram is also sent by the passive if the T_CTp expires. T_CTa and T_CTp are reset when a Connection Test message is seen.

- Break: The Break telegram is used by the passive ECU to stop a transfer of data from an active ECU. After receiving a break, the active ECU should send a Data telegram with the End of Message bit set and a Ack request.

- Disconnect: The Disconnect telegram is used by the active ECU to cancel (disconnect) a Transport Protocol channel.

TPDU message telegram types and there message size:

TABLE 3 - GENERAL TPDU FORMATS

TPDU Type	Acronym	TPDU Bytes							
		0	1	2	3	4	5	6	7
Data	DT	TPCI1	D / -	D / -	D / -	D / -	D / -	D / -	D / -
Acknowledge	Ack	TPCI1	-	-	-	-	-	-	-
Connection set-up	CS	TPCI1 = 0xA0	TPCI2	T1	T2	T3	T4	-	-
Connection ack.	CA	TPCI1 = 0xA1	TPCI2	T1	T2	T3	T4	-	-
Connection test	CT	TPCI1 = 0xA3	-	-	-	-	-	-	-
Break	BR	TPCI1 = 0xA4	-	-	-	-	-	-	-
Disconnect	DC	TPCI1 = 0xA8	-	-	-	-	-	-	-

‘-‘ Bytes are not sent; the length of the telegram must be adapted for each TPDU type.

5.2.1 Control Bytes

The control byte (TPCI1) has the following structure:

TABLE 4 - TPCI BYTE 1

TPDU Type	Acronym	TPCI Byte 1							
		7	6	5	4	3	2	1	0
Data	DT	0	0	AR	EOM	SN			
Acknowledge	Ack	1	0	RS	1	SN			
Connection set-up	CS	1	0	1	0	0	0	0	0
Connection ack.	CA	1	0	1	0	0	0	0	1
Connection test	CT	1	0	1	0	0	0	1	1
Break	BR	1	0	1	0	0	1	0	0
Disconnect	DC	1	0	1	0	1	0	0	0

AR - Acknowledge request: 0 = Request; 1 = No Request
EOM - End of Message: 0 = False; 1 = True
RS - Receive status: 0 = Receiver not ready; 1 = Receiver Ready If Receiver is not ready, insert a delay of T_wait before sending the next Data telegram.
SN - Sequence number: 0x0 through 0xF. If an Ack is requested, the SN for the Ack is the SN from the data message plus 1.

The control byte (TPCI2) has the following structure:

TABLE 5 - TPCI BYTE 2

TPDU Type	Acronym	TPCI Byte 2							
		7	6	5	4	3	2	1	0
Connect set-up	CS	0	0	0	0	BS			
Connect ack.	CA	0	0	0	0	BS			

BS - Block Size: Number of directly following telegrams after which an acknowledgement must be requested. Value range:0 < BS < 16. Use the smaller of either the Set-up BS or the Ack BS. Each ECU must always send the current maximal possible value for the transfer.

Sending priorities of the TPDU types

Table 4 lists the sending priorities of the individual TPDU types. In the case that different types have to be sent at the same time, the type with the highest priority must be transferred on the bus. The smaller the number, the higher is the priority.

TABLE 6 - SENDING PRIORITY

TPDU Type	Priority
DT	4
Ack	3
CS	4
CA	1
CT	1
BR	2
DC	4

5.2.2 Dynamic Transport Protocol Timing Parameters

Timings and block sizes of the channel are established with the connection set-up / acknowledge message. The transport layer must reset all variables (SN, counter, timer) after the exchange of connection set-up / acknowledge telegrams.

Description of the parameters:

T1 Time-out used by this ECU for received telegrams
T2 not used at this time; always set to ∝ (0xFF)
T3 minimum time the sending ECU should put between consecutive telegrams being sent to this ECU.
T4 not used at this time; always set to ∝ (0xFF)

Bytes T1, T2, T3 and T4 are constructed as follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Time Base		Time					

Time Base 00 = 100 µsec
01 = 1 msec
10 = 10 msec
11 = 100 msec
Time: 0..63

The time is calculated by multiplying time base by the time

Special cases:
T1 = 0xFF No time-out
T3 = 0x00 Immediate successively following telegrams are permissible

For setting the timings that must be used, the following conditions must be observed because of the sending priorities:

T1 > 4 x T3

5.2.3 Static Transport Protocol Parameters

For all transport protocol connections the following parameters are statically established:

TABLE 7 - STATIC TRANSPORT PROTOCOL PARAMETERS

Description	**Parameter Name**	**Value**
Connection Test		
Active ECU Connection Test timeout (runs concurrent with other actions)	T_CTa	1000 ms
Passive ECU Connection Test timeout (time-out waiting for the Connection Test telegram)	T_CTp	1050 ms
Maximum Repeats of the connection test telegram	MNCT	5
Request / Response		
Maximum number acceptances of telegram not ready requests within a block size	MNTB	5
Maximum repeats of acknowledge requests	MNT	2
Acknowledge with Receiver not ready		
Delay timer used to delay next telegram if Receiver Not Ready bit set in Ack Telegram	T_Wait	100 ms

Repetition Count of ‘n’ means that the applicable telegram was sent n+1 in total.

5.2.4 Transport Protocol Error Handling

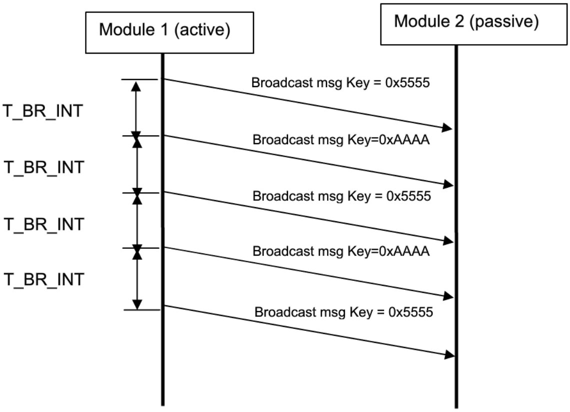
TABLE 8 - TRANSPORT PROTOCOL ERROR HANDLING

Condition	**Action**
Passive ECU receives a Data telegram with an unexpected SN	Send Ack with the SN that is expected.
Active ECU receives Ack with request for previous SN telegram	Send the asked for Data Telegram up to MNTB times then error by closing the channel (send Disconnect telegram).
Active ECU time-out on T1 when expecting a Ack response	Repeat the last Data Telegram for up to MNTC times then error by closing the channel (send Disconnect telegram)
Passive ECU T_CTp timeout waiting for Connection Test telegram	Count each successive occurrence, when MNCT is exceeded, close the channel (send Disconnect telegram)

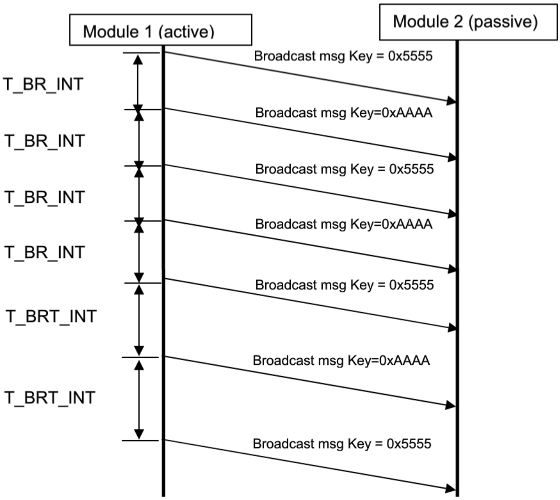
6. EXAMPLES

6.1 CAN-Telegram Examples

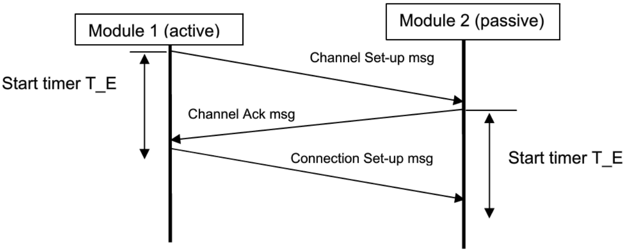
6.1.1 Broadcast without re-trigger



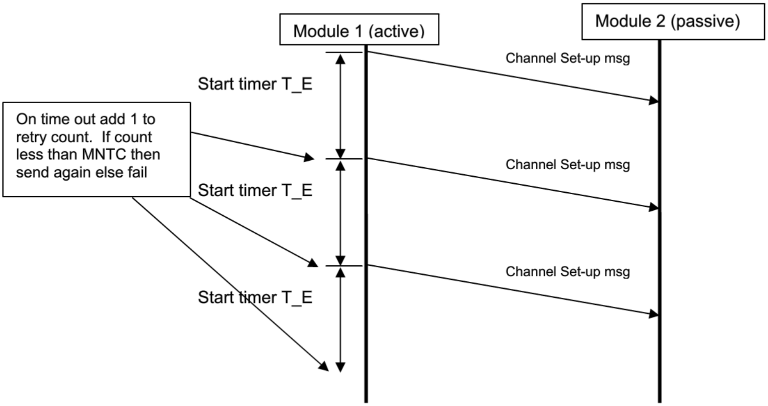
6.1.2 Broadcast with re-trigger



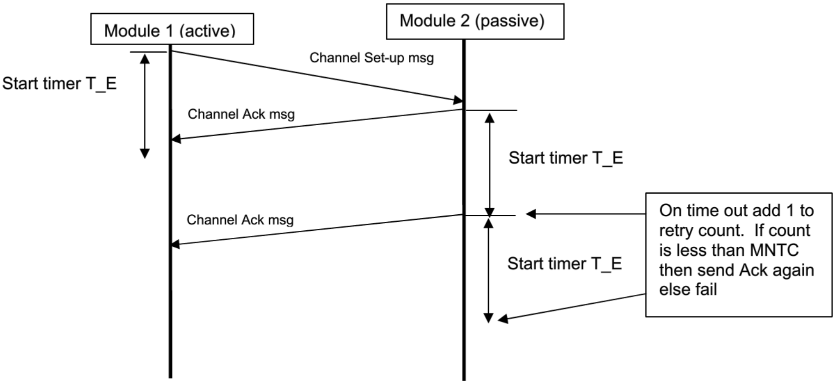
6.1.3 Channel Set-up with Ack



6.1.4 Channel Set-up missing Ack

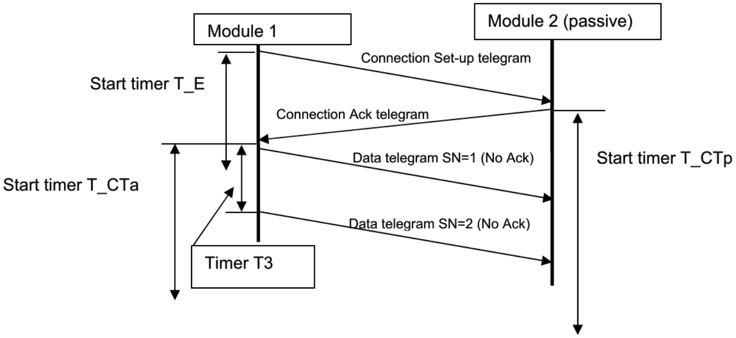


6.1.5 Channel Set-up with Ack missing Transport Protocol Connection Set-up

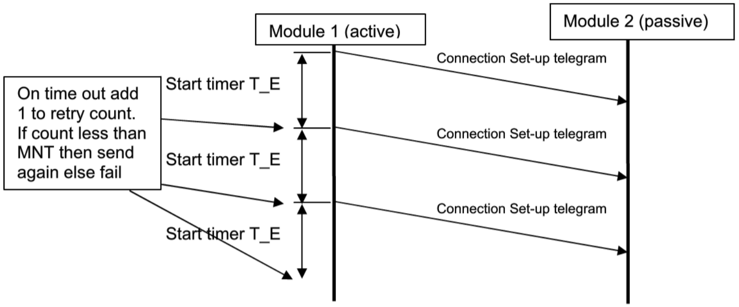


6.2 Transport Protocol Examples

6.2.1 Connection Set-up with Ack

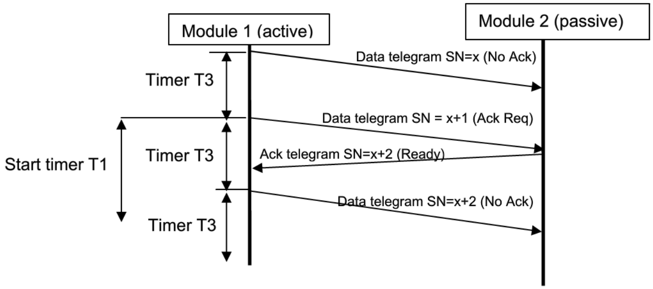


6.2.2 Connection Set-up missing Ack

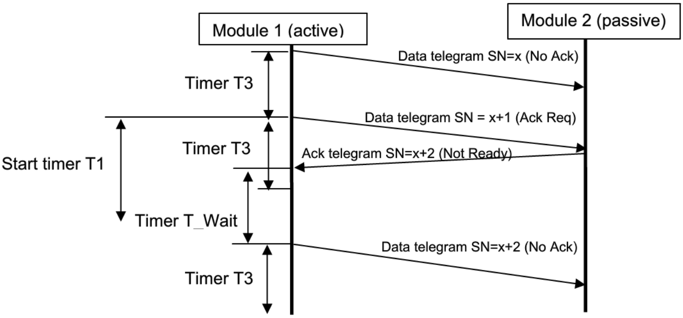

Module 1 (active)
Module 2 (passive)
Connection Set-up telegram
Start timer T_E
On time out add 1 to retry count. If count less than MNT then send again else fail
Connection Set-up telegram
Start timer T_E
Connection Set-up telegram
Start timer T_E

6.2.3 Sending Data with Acknowledge request - ready response

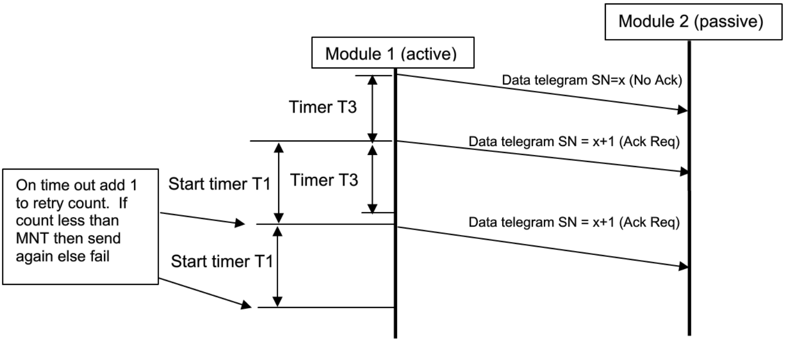
NOTE: The Connection Test timers T_CTa and T_CTp are running through all of the following diagrams. It is not shown in order to simplify the diagrams.


Module 1 (active)
Module 2 (passive)
Data telegram SN=x (No Ack)
Timer T3
Data telegram SN = x+1 (Ack Req)
Start timer T1
Timer T3
Ack telegram SN=x+2 (Ready)
Data telegram SN=x+2 (No Ack)
Timer T3

6.2.4 Sending Data with Acknowledge request not ready

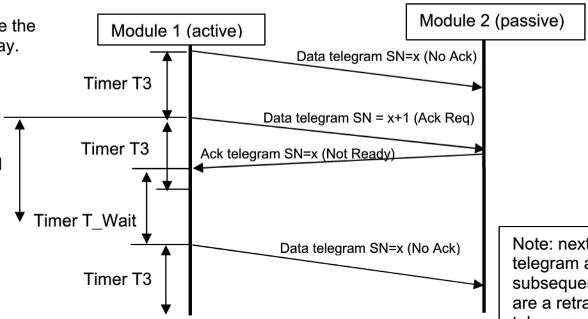

Module 1 (active)
Module 2 (passive)
Data telegram SN=x (No Ack)
Timer T3
Data telegram SN = x+1 (Ack Req)
Start timer T1
Timer T3
Ack telegram SN=x+2 (Not Ready)
Timer T_Wait
Data telegram SN=x+2 (No Ack)
Timer T3

6.2.5 Sending Data with Acknowledge request with no Ack


Module 2 (passive)
Module 1 (active)
Data telegram SN=x (No Ack)
Timer T3
Data telegram SN = x+1 (Ack Req)
On time out add 1 to retry count. If count less than MNT then send again else fail
Start timer T1
Timer T3
Data telegram SN = x+1 (Ack Req)
Start timer T1

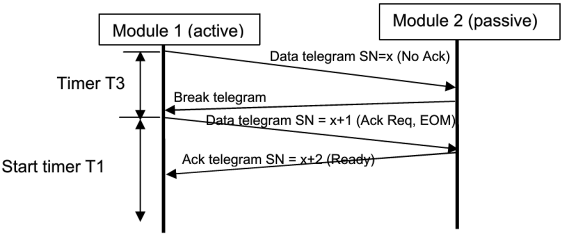
6.2.6 Sending Data with Acknowledge request, Receiver not ready and retransmit block previous block

The “Not Ready” Ack cause the Insertion of the T_Wait delay.

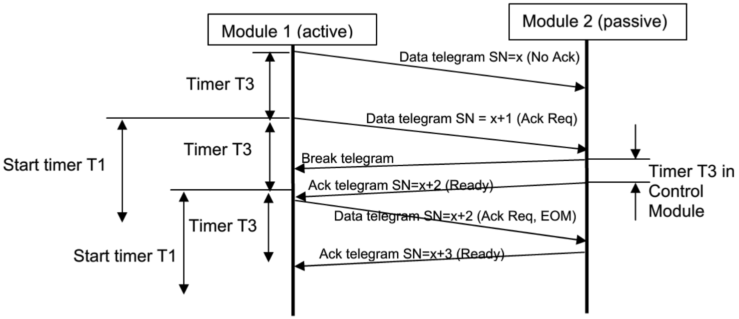

Module 1 (active)
Module 2 (passive)
Data telegram SN=x (No Ack)
Timer T3
Data telegram SN = x+1 (Ack Req)
Start timer T1
Timer T3
Ack telegram SN=x (Not Ready)
Timer T_Wait
Data telegram SN=x (No Ack)
Timer T3

Note: next Data telegram and subsequent telegrams are a retransmit of telegrams starting at SN = x.

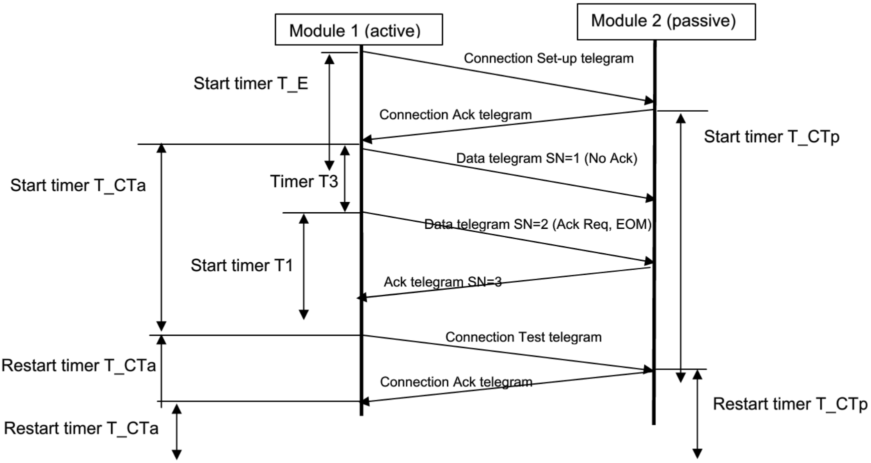
6.2.7 Break in between Data telegrams without Ack request


Module 1 (active)
Module 2 (passive)
Data telegram SN=x (No Ack)
Timer T3
Break telegram
Data telegram SN = x+1 (Ack Req, EOM)
Ack telegram SN = x+2 (Ready)
Start timer T1

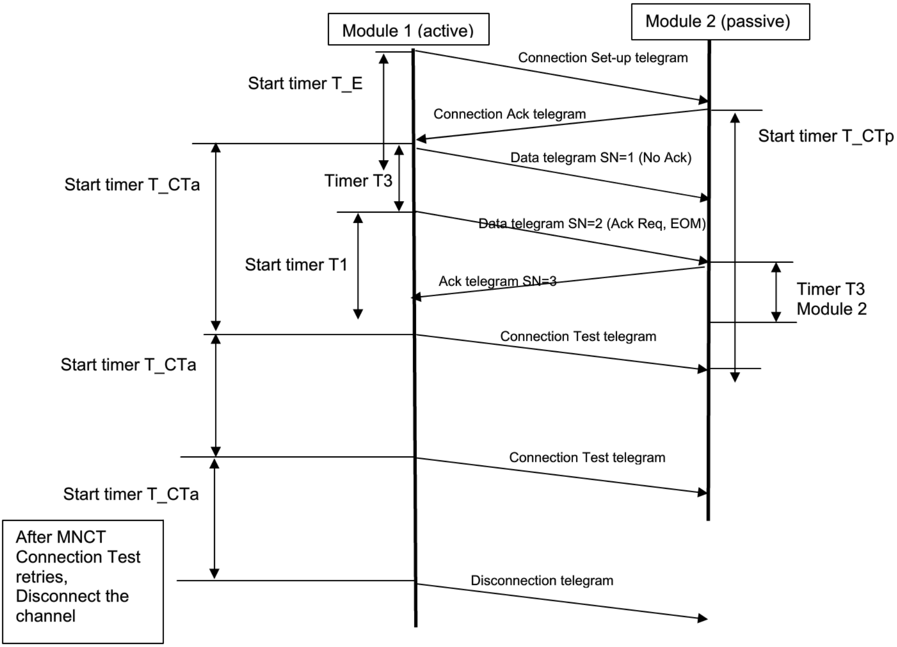
6.2.8 Break in between Data telegrams with Ack request


Module 1 (active)
Module 2 (passive)
Data telegram SN=x (No Ack)
Timer T3
Data telegram SN = x+1 (Ack Req)
Start timer T1
Timer T3
Break telegram
Timer T3 in Control Module
Ack telegram SN=x+2 (Ready)
Data telegram SN=x+2 (Ack Req, EOM)
Timer T3
Start timer T1
Ack telegram SN=x+3 (Ready)

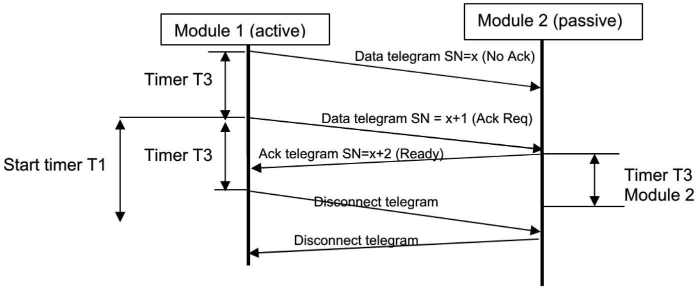
6.2.9 Connection Test telegram with Connection Ack



6.2.10 Connection Test telegram missing Connection Ack



6.2.11 Disconnect telegram



7. NOTES

7.1 Marginal Indicia

A change bar (l) located in the left margin is for the convenience of the user in locating areas where technical revisions, not editorial changes, have been made to the previous issue of this document. An (R) symbol to the left of the document title indicates a complete revision of the document, including technical revisions. Change bars and (R) are not used in original publications, nor in documents that contain editorial changes only.

PREPARED BY THE SAE J2534/2 OPTIONAL FEATURES TASK FORCE OF THE
VEHICLE E/E SYSTEM DIAGNOSTIC STANDARDS COMMITTEE