INTERNATIONAL STANDARD

ISO 14230-2

First edition
1999-03-15

Road vehicles — Diagnostic systems — Keyword Protocol 2000 —

Part 2: Data link layer

Véhicules routiers — Systèmes de diagnostic — Protocole «Keyword 2000» —

Partie 2: Couche de liaison de données

Reference number
ISO 14230-2:1999(E)

Contents

Page

International Organization for Standardization
Case postale 56 • CH-1211 Genève 20 • Switzerland
Internet iso@iso.ch

Printed in Switzerland

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO 14230-2 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, subcommittee SC 3, *Electrical and electronic equipment*.

ISO 14230 consists of the following parts, under the general title *Road vehicles — Diagnostic systems — Keyword Protocol 2000* :

— *Part 1: Physical layer*

— *Part 2: Data link layer*

— *Part 3: Application layer*

— *Part 4: Requirements for emissions-related systems*

Annex A forms an integral part of this part of ISO 14230. Annexes B and C are for information only.

Introduction

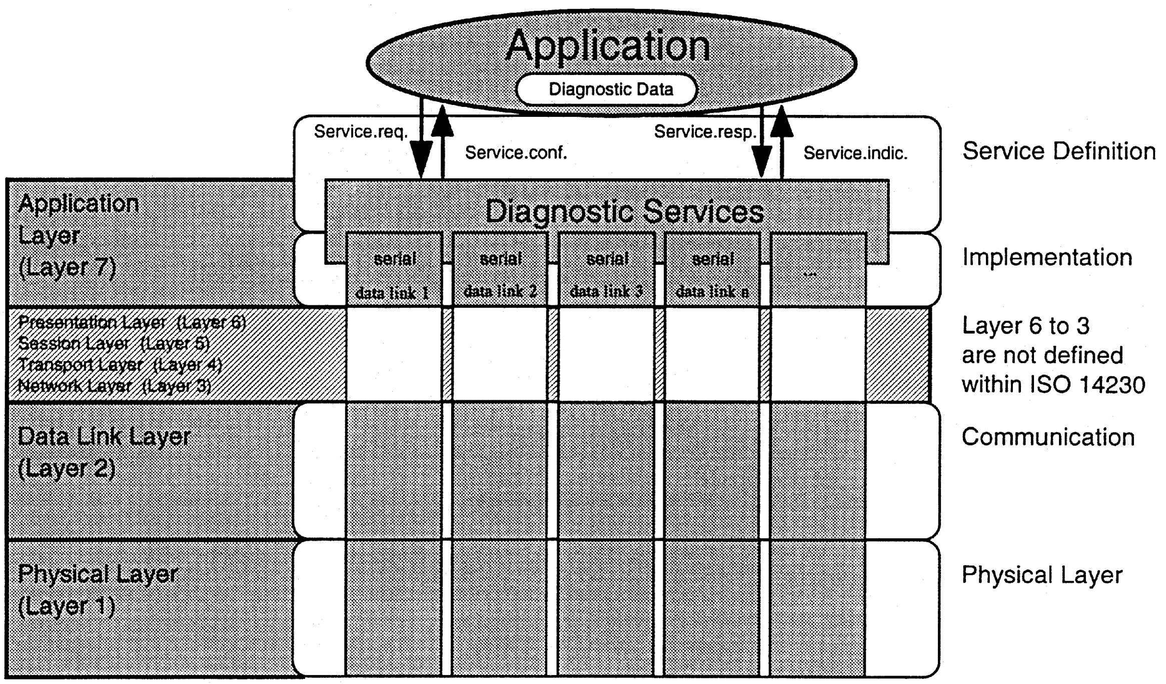
ISO 14230 has been established in order to define common requirements for diagnostic systems implemented on a serial data link.

To achieve this, it is based on the Open System Interconnection (OSI) Basic Reference Model in accordance with ISO 7498 which structures communication systems into seven layers. When mapped on this model, the services used by a diagnostic tester and an Electronic Control Unit (ECU) are broken into:

— diagnostic services (layer 7);

— communication services (layers 1 to 6),

in accordance with figure 1.



Example of serial data links: KWP2000, VAN, CAN, J1850, etc.

Figure 1 — Mapping of diagnostic services on OSI Model

Road vehicles — Diagnostic systems — Keyword Protocol 2000 —

Part 2:
Data link layer

1 Scope

This part of ISO 14230 specifies common requirements of diagnostic services which allow a tester to control diagnostic functions in an on-vehicle Electronic Control Unit (for example, electronic fuel injection, automatic gearbox, antilock braking system, etc.) connected on a serial data link embedded in a road vehicle.

It specifies only layer 2 (data link layer). Included are all definitions which are necessary to implement the services (described in ISO 14230-3) on a serial link (described in ISO 14230-1). Also included are some communication services which are needed for communication/session management and a description of error handling.

This part of ISO 14230 does not specify the requirements for the implementation of diagnostic services.

The physical layer may be used as a multiuser-bus, so a kind of arbitration or bus management is necessary. There are several proposals which are not part of this part of ISO 14230. The car manufacturers are responsible for the correct working of bus management.

Communication between ECUs are not part of this part of ISO 14230.

The vehicle diagnostic architecture of this part of ISO 14230 applies (see figure 2) to

— a single tester that may be temporarily or permanently connected to the on-vehicle diagnostic data link, and

— several on-vehicle electronic control units connected directly or indirectly.

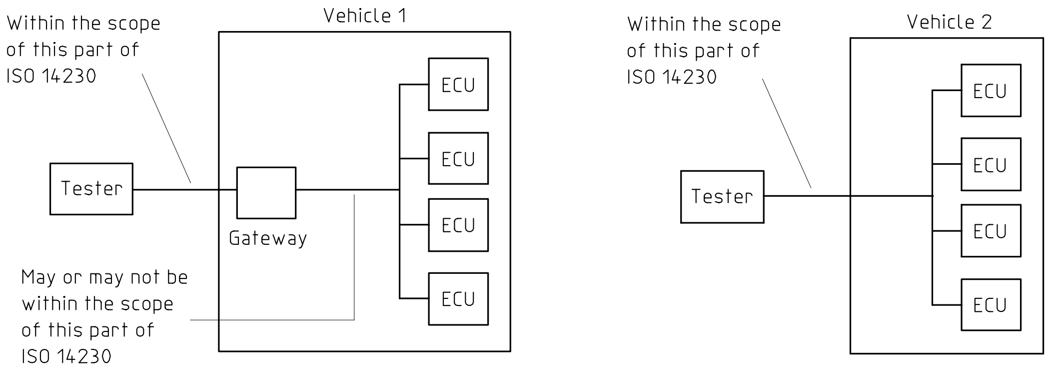


Figure 2 — Vehicle diagnostic architecture

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO 14230. All the time of publication, the editions were indicated were valid. All standards are subject to revision, and parties to agreement based on this part of ISO 14230 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 9141:1989, *Road vehicles — Diagnostic systems — Requirements for interchange of digital information.*

ISO 9141-2:1994, *Road vehicles — Diagnostic systems — Part 2: CARB requirements for interchange of digital information.*

ISO 14230-3:1999, *Road vehicles — Diagnostic systems — Keyword Protocol 2000 — Part 3: Application layer.*

ISO 14230-4:1999, *Road vehicles — Diagnostic systems — Keyword Protocol 2000 — Part 4: Requirements for emission related systems.*

SAE J 1979: 1996, *E/E diagnostic test modes.*

3 Physical topology

Keyword Protocol 2000 is a bus concept. Figure 3 shows the general form of this serial link.

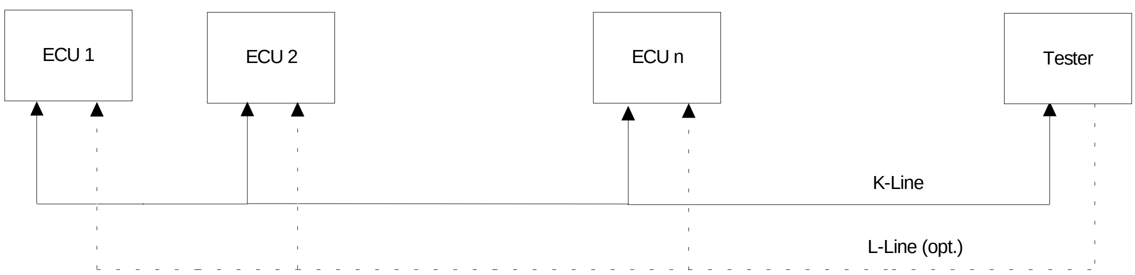


Figure 3 — Topology

The K-line is used for communication and initialization; the L-line (optional) is used for initialization only. Special cases are node-to-node-connection, which means there is only one ECU on the line which also can be a bus converter.

4 Message structure

4.1 General

The message structure consists of three parts:

— header;

— data bytes;

— checksum.

See figure 4.

Header and checksum bytes are described in this part of ISO 14230. The area of data bytes always begins with a Service Identification.

The data bytes and their use are described in ISO 14230-3 and this part of ISO 14230.

Header				Data bytes		Checksum
Fmt	Tgt[1)]	Src[1)]	Len[1)]	SId[2)]	. . Data[2)] . .	CS
max . 4 byte				max. 255 byte		1 byte

1) bytes are optional, depending on format byte.

2) Service Identification, part of data bytes.

NOTE — The shaded area (header, checksum) are described in this part of ISO 14230.

Figure 4 — Message structure

4.2 Header

The header consists of a maximum of 4 bytes. A format byte includes information about the form of the messages, target and source address bytes are optional for use with multinode connections, an optional separate length byte allows message lengths up to 255 bytes.

4.2.1 Format byte

The format byte contains 6 bit length information and 2 bit address mode information. The tester is informed about use of header bytes by key bytes (see 4.2.2).

A1	A0	L5	L4	L3	L2	L1	L0

where

A1 and A0 define the form of header which will be used by the message in accordance with table 1;

L5...L0 define the length of a message from the beginning of the data fields (ServiceIdentification byte included) to checksum byte (not included). A message length of 1 to 63 bytes is possible. If L0 to L5 = 0 then the additional length byte is included (see 4.2.5).

Table 1 — Header message form

A1	A0	Mode
0	0	no address information
0	1	Exception mode (CARB)
1	0	with address information, physical addressing
1	1	with address information, functional addressing

A.1,A.0=01 (CARB mode) is an exception mode. The CARB mode is not specified in this part of ISO 14230. CARB uses format bytes $68 (0110 1000) and $48 (0100 1000). For more details refer to ISO 9141-2 and SAE J1979.

4.2.2 Target address byte

This is the target address for the message and is always used together with the source adress byte. The target address in the request messages sento to the ECU may be a physical or a functional address. The target address in the response messages sent to the tester shall be the physical address of the tester. Physical addresses may be the 5 baud address byte (see annex A) or addresses according to SAE J 2178-1 (see annex B). The target address byte is optional and only necessary on multimode bus topologies. For node-to-node connections it may be omitted. For CARB messages this byte is defined in ISO 9141-2 or ISO 14230-4.

4.2.3 Source address byte

This is the address of the transmitting device. It shall be a physical address. There are the same possibilities for the values as described for physical target address bytes. Addresses for testers are listed in SAE J2178-1 (see annex B). This byte is optional (always used together with target address byte) and only necessary on multinode bus topologies. For node-to-node connections it may be omitted.

4.2.4 Length byte

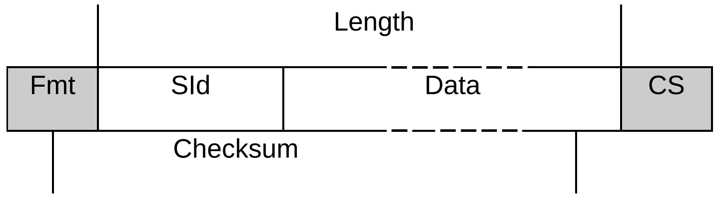
This byte is provided if the length in the header byte (L0 to L5) is set to 0 as shown in table 2. It allows the user to transmit messages with data fields longer then 63 bytes. With shorter messages it may be omitted. This byte defines the length of a message from the beginning of the data field (service identification byte included) to checksum byte (not included). A data length of 1 byte to 255 bytes is possible. The longest message consists of a maximum of 260 bytes. For messages with data fields of less than 64 bytes there are two possibilities: length may be included in the format byte or in the additional length byte. An ECU does not need to support both possibilities, the tester is informed about the capability of an ECU through the keybytes (see 6.1.2.1).

Table 2 — Presence of a length byte

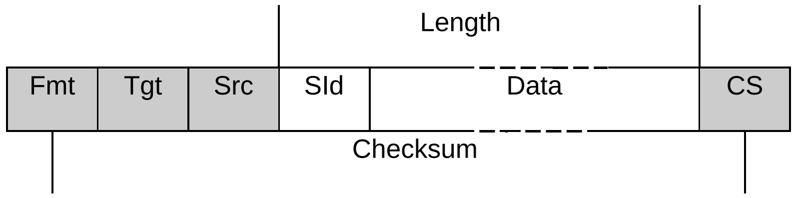
Length	Length provided in	
	Fmt byte[1)]	Length byte
< 64	XX00 0000	present
< 64	XXLL LLLL	not present
≥ 64	XX00 0000	present
1) XX : 2 bits address mode information (see 4.2.1) LL LLLL : 6 bits length information.		

4.2.5 Use of header bytes

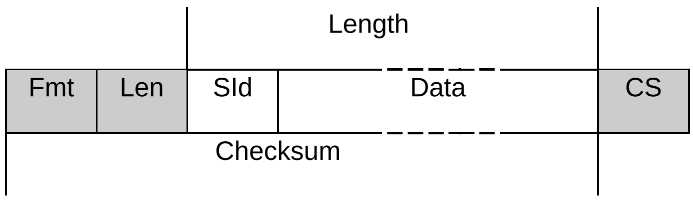
With the above definitions there are four different forms of message. These are shown diagramatically in figure 5.



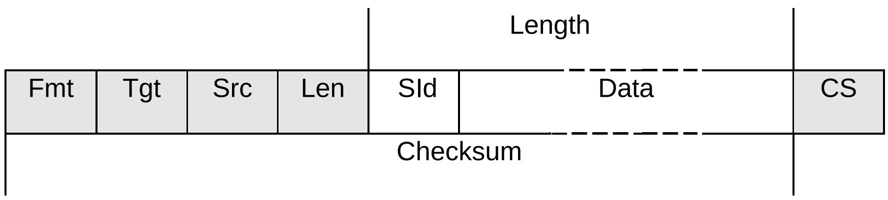
a) Header with address information, no additional length byte



b) Header with address information, no additional length byte



c) Header without address information, additional length byte



d) Header with address information, with additional length byte

Fmt	Format byte	SId	Service Identification Byte
Tgt	Target address (optional)	Data	(depending on service)
Src	Source address (optional)	CS	Checksum byte
Len	additional length byte (optional)		

NOTE — The unshaded area is defined in ISO 14230-3.

Figure 5 — Header messages

4.3 Data bytes

The data field may contain up to 63 or up 255 bytes of information, depending on the use of length information. The first byte of the data field is the Service Identification Byte. It may be followed by parameters and data depending on the selected service. These bytes are defined in ISO 14230-3 (for diagnostic services) and in clause 5 (for communication services).

4.4 Checksum byte

The checksum byte (CS) inserted at the end of the message block is defined as the simple 8 bit sum series of all bytes in the message, excluding the checksum.

If the message is

$\langle 1\rangle \langle 2\rangle \langle 3\rangle \ldots \langle N\rangle, \langle CS\rangle$

the two following cases may occur:

when $\langle i\rangle$ $(1 \le i \le N)$ is the numerical value of the i^{th} message byte, then :

$\langle CS\rangle = \langle CS\rangle_N$

when $\langle CS\rangle_i$ $(2 \le i \le N)$:

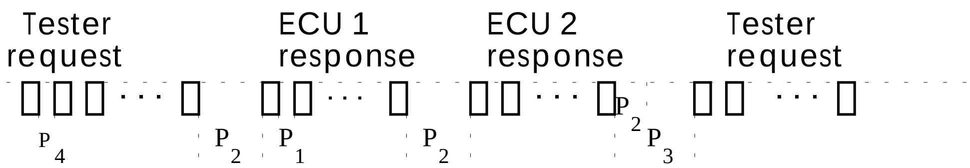
when $\langle CS\rangle_i = \{ \langle CS\rangle_{i-1} + \langle i\rangle \}$ mod256 et
$\langle CS\rangle_1 = \langle 1\rangle$

Additional security may be included in the data field as defined by the manufacturer.

4.5 Timing

4.5.1 Value entering

During normal operation the timing parameters as shown in figure 6 are relevant.



Value	Description
P1	Inter byte time for ECU response
P2	Time between tester request and ECU response or two ECU responses
P3	Time between end of ECU responses and start of new tester request
P4	Inter byte time for tester request

Figure 6 — Timing

There are two sets of default timing parameters :

a) one set for normal functional and physical addressed communication. Longer timings are necessary to allow any technics of bus management;

b) one set restricted to physical addressing to allow faster communication.

The tester is informed about the capability of an ECU through the keybytes (see 5.2.4.1).

Timing parameters may be changed with the communication service "AccessTimingParameters" (see 5.4).

Users shall take note of limits listed below and the following restrictions:

$P3_{min} > P4_{min}$

$Pi_{min} < Pi_{max}$ for $i = 1, ..., 4$

There may be further restrictions, when the tester and listening ECUs detect the end of a message by timeout. In this case the following restrictions are valid:

$P2_{min} > P4_{max}$

$P2_{min} > P1_{max}$

In case of functional addressing, i.e. that there may be more than one response to one request, further restrictions may be added.

It is in the designers' responsibility to ensure proper communication in the case of changing the timing parameters from the default values.They shall also ensure that the chosen communication parameters are possible for all ECUs which participate in the session.

The possible values depend on the capabilities of the ECU. In some cases the ECU may need to leave its normal operation mode to switch over to a session with different communication parameters.

Tables 3 and 4 show the timing parameters which are used as default, the limits within which they can be changed and the resolution which may be used to set a new value (with the communication service AccessTimingParameter: see 5.4).

Table 3 — Normal Timing Parameters Set (for functional and physical addressing)

Values in milliseconds

Timing Parameter	Minimum values			Maximum values		
	Lower limit	Default	Resolution	Default	Upper limit	Resolution
P1	0	0	-	20	20	-
P2	0	25	0,5	50	See table 5	
P3	0	55	0,5	5 000	∞ ($FF)	250
P4	0	5	0,5	20	20	-

Table 4 — Extended Timing Parameters Set (for physical addressing only)

Values in milliseconds

Timing Parameter	Minimum values			Maximum values		
	Lower limit	Default	Resolution	Default	Upper limit	Resolution
P1	0	0	-	20	20	-
P2	0	25	0,5	50	See table 5	
P3	0	55	0,5	5 000	∞ ($FF)	250
P4	0	5	0,5	20	20	-

Table 5 — P2max Timing Parameter calculation

Hex. value	Resolution	Maximum value ms	Maximum value calculation method ms
01 to F0	**25**	**25 to 6 000**	**(hex. value) x (Resolution)**
F1	see maximum value calculation method	6 400	(low nibble of hex. value) x 256 x 25 Example of $FA : ($0A x $0100) x 25 = 64 000
F2		12 800	
F3		19 200	
F4		25 600	
F5		32 000	
F6		38 400	
F7		44 800	
F8		51 200	
F9		57 600	
FA		64 000	
FB		70 400	
FC		76 800	
FD		83 200	
FE		83 600	
FF	-	∞	Not applicable
The P2max timing parameter value shall always be a single byte value in the AccessTimingParameter service. The timing modifications shall be activated by implementation of the AccessTimingParameter service.			

Proposed $P2_{max}$ timing parameter calculation method (values > 6 000 ms):

The $P2_{max}$ timing parameter calculation uses 25 ms resolution in the range of $01 to $F0. Beginning with $F1, a different calculation method shall be used by the server and the client in order to reach $P2_{max}$ timing values greater than 6 000 ms.

Calculation Formula $P2_{max}$ values > $F0

Calculation_Of_$P2_{max}$ = (low nibble of $P2_{max}$) $\times$ 256 $\times$ 25 (ms)

The P2max timing parameter value shall always be a single byte value in the AccessTimingParameter service. The timing modifications shall be activated by implementation of the AccessTimingParameter service.

4.5.2 Timing exceptions

The extended P2 timing window is a possibility for (a) server(s) to extend the time to respond on a request message. A P2max timing exception is only allowed with the use of one or multiple negative response message(s) with response code $78 (RequestCorrectlyReceived-ResponsePending) by the server(s). This response code shall only be used by a server in case it cannot send a positive or negative response message based on the client's request message within the active P2 timing window. This response code shall manipulate the P2max timing parameter value in the verver and the client. The P2max timing parameter is set to the value (in ms) of the P3max timing parameter. The client shall remain in the receive mode. The server(s) shall send multiple negative response messages with the negative response code $78 if required.

As soon as the server has completed the task (ourtine) initiated by the request message it shall send either a positive or negative response message (negative response message with a response code other than $78) based on the last request message received. When the client has received the response message which has been preceeded by the negative response message(s) with response code $78, the client and the server shall reset the P2max timing parameter to the previous timing value. The client shall not repeat the request message after the reception of a negative response message with response $78 .

5 Communication services

5.1 General

Some services are necessary to establish and maintain communication. Those are not diagnostic services because they do not appear on the application layer. They are described in the same formal way and with the same conventions (CVT) as the Diagnostic Services (see ISO 14229): a service table and a verbal description of the service procedure. The existence of parameters is shown as follows:

— mandatory: M,

— selectable: S,

— conditional: C,

— user-optional: U.

A description of implementation on the physical layer of Keyword Protocol 2000 is added.

In general services are not mandatory: only StartCommunication Service shall be implemented.

The StartCommunication Service and the AccessTimingParameters Service are used for starting a diagnostic communication. In order to perform any diagnostic service, communication shall be initialized and the communication parameters need to be appropriate to the desired diagnostic mode. A chart describing this is shown in figure 7.

5.2 StartCommunication Service

5.2.1 Service definition

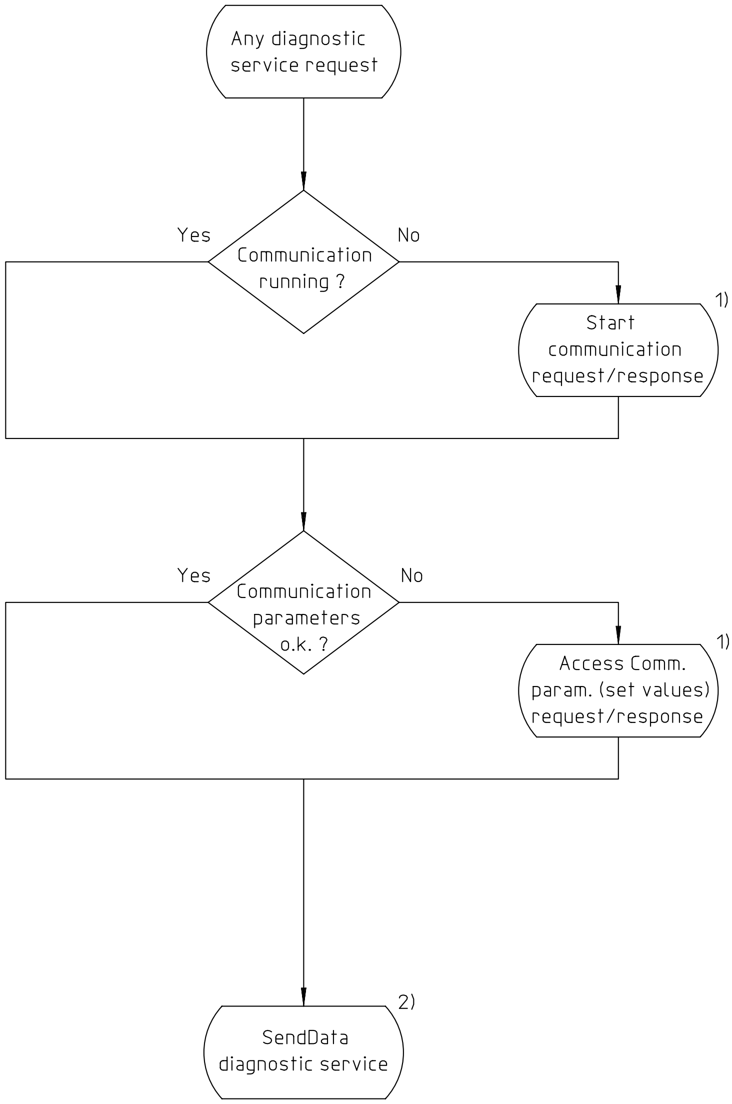
The purpose of this KWP 2000 communication layer service is to initialize the communication link for the exchange of diagnostic data.

5.2.2 Service table

See table 6.

Table 6 — StartCommunication Service

StartCommunication Request	M
Initialization Mode Identifier[1)]	M
Target Initialization Address	M
Source Initialization Address	C1[2)]
StartCommunication Positive Response Target Address Source Adress Key bytes Keybytes	M C2[3)] C2[3)] M2 M2
1) The way of initialization is determined by the Initialization Mode Identifier, the value of this parameter may be CARB-initialization, 5-baud initialization or fast initialization. 2) C1 : Source initialization address is added if initialization Mode Identifier = Fast Initialization. 3) C2 : Target and source address are added if addresss information is used in the header (three or four byte header).	



1) Communication Service

2) Diagnostic Service

Figure 7 — Use of communication services

5.2.3 Service procedure

Upon receiving a StartCommunication indication primitive, the ECU shall check if the requested communication link can be initialized under the present conditions. Valid conditions for the initialization of a diagnostic communication link are described in 5.3.2.

Then the ECU shall perform all actions necessary to initialize the communication link and send a StartCommunication response primitive with the Positive Response parameters selected.

If the communication link cannot be initialized for any reason, the ECU shall maintain its normal operation (see 6.1).

5.2.4 Implementation

The StartCommunication Service is used to initialize a communication on the K-line. There are different possibilities to initialize:

a) CARB initialization;

b) 5-Baud initialization;

c) fast initialization.

Figure 8 shows three possibilities and the ECU status after each kind of initialization. After finishing the initialization, the ECUs are in the same status, regardless of the initialization mode:

— all communication parameters are set to default values according to the key bytes;

— ECU is waiting for the first request of the tester for a time period of P3;

— ECU is in the default diagnostic mode (i.e., it has a well defined functionality).

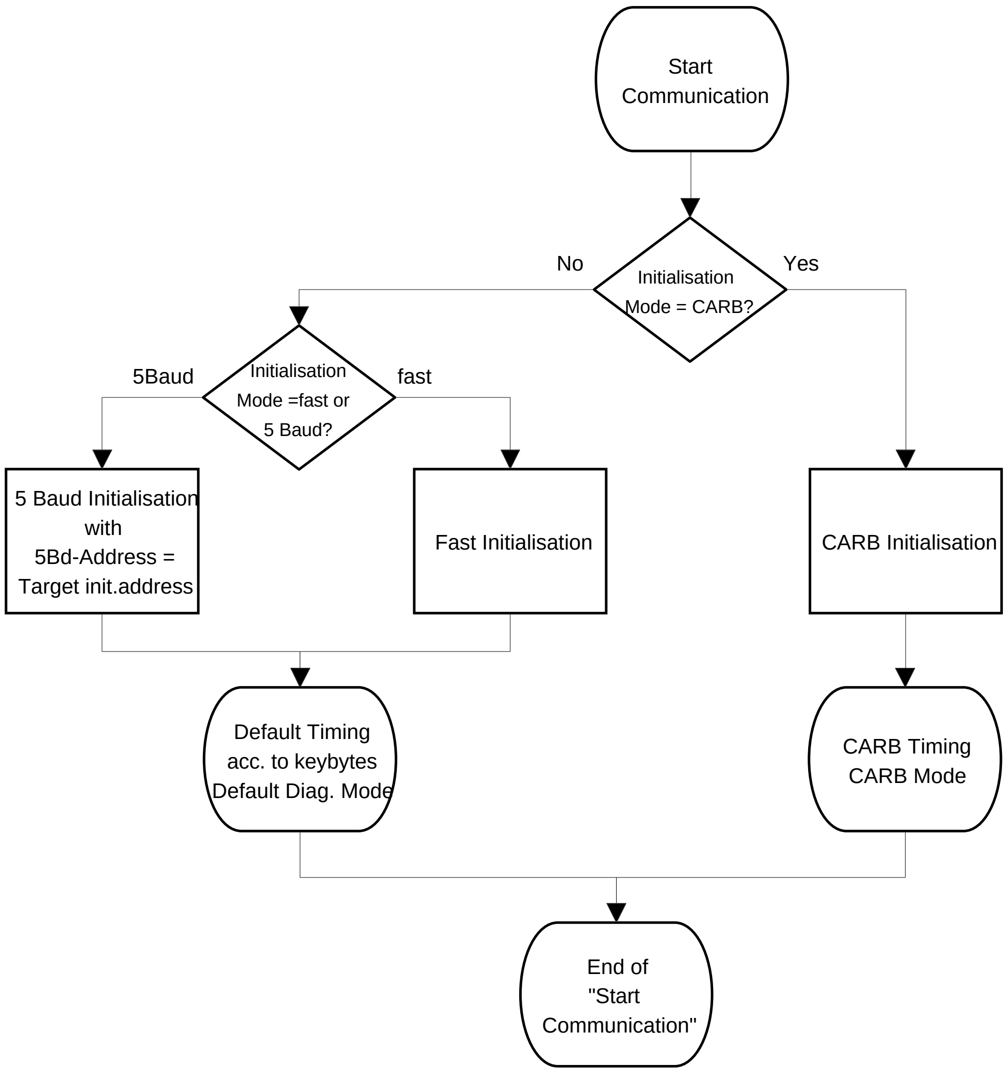


Figure 8 — Initialization modes

There are general facts that are common to all modes of initialization:

— prior to any activity there shall be a bus-idle time;

— then the tester sends an initialization pattern;

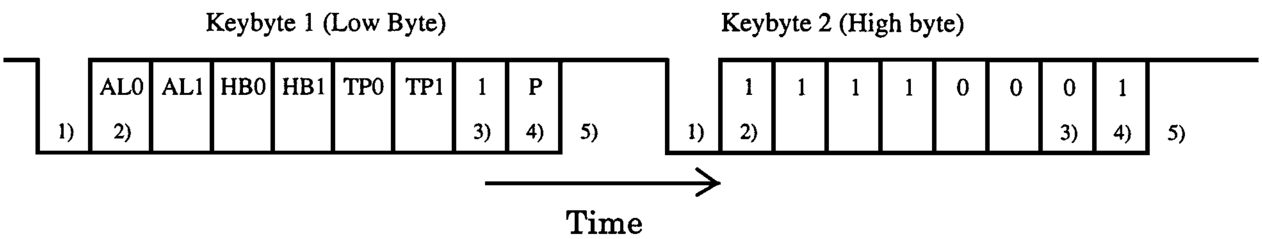
— all information which is necessary to establish communication is contained in the response of the ECU.

5.2.4.1 Key bytes

With these bytes an ECU informs the tester about the supported header, timing and length information. So an ECU does not necessarily have to support all possibilities.

The decoding of the key bytes is defined in ISO 9141.

Graphical representation of the keybytes is given in figure 9, and meanings of each of their bit values in table 7. Table 8 gives possible keybyte values.



1) Binary start bit

2) Least significant bit

3) Most significant bit

4) Eqality bit

5) Binary stop bit

Figure 9 — Keybytes

Table 7 — Meaning of bit values in keybytes

Bit	Value	
	0	1
AL0	length inf. in format byte not supported	length inf. in format byte supported
AL1	add. length byte not supported	add. length byte supported
HB0	1 byte header not supported	1 byte header supported
HB1	Tgt/Src address in heater not supported	Tgt/Src address in heater supported
TP0 1)	normal timing parameter set	extended timing parameter set
TP1 1)	extented timing parameter set	normal timing parameter set
1) Only TP0, TP1 = 0,1 and 1,0 allowed.		

Table 8 — Possible values of Keybytes

Keybytes				Supported		Time
Binary		Hex	Dec.1)	Length information	Type of header	
KB2	KB1					
1000 1111	1101 0000	$8FD0	2000	2)		
1000 1111	1101 0101	$8FD5	2005	format byte	Header without address information	Extended timing
1000 1111	1101 0110	$8FD6	2006	additional length byte		
1000 1111	0101 0111	$8F57	2007	both modes possible		
1000 1111	1101 1001	$8FD9	2009	format byte	Header with target and source address information.	
1000 1111	1101 1010	$8FDA	2010	additional length byte		
1000 1111	0101 1011	$8F5B	2011	both modes possible		
1000 1111	0101 1101	$8F5D	2013	format byte	both types of header supported	
1000 1111	0101 1110	$8F5E	2014	additional length byte		
1000 1111	1101 1111	$8FDF	2015	both mode possible		
1000 1111	1110 0101	$8FE5	2021	format byte	Header without address information	normal timing
1000 1111	1110 0110	$8FE6	2022	additional length byte		
1000 1111	0110 0111	$8F67	2023	both mode possible		
1000 1111	1110 1001	$8FE9	2025	format byte	Header with target and source address information	
1000 1111	1110 1010	$8FEA	2026	additional length byte		
1000 1111	0110 1011	$8F6B	2027	both modes possible		
1000 1111	0110 1101	$8F6D	2029	format byte	Both types of header supported	
1000 1111	0110 1110	$8F6E	2030	additional length byte		
1000 1111	1110 1111	$8FEF	2031	both modes possible		

1)To calculate the decimal value, clear the parity bit of both keybytes and then multiply keybyte 2 by 2^7 and add keybyte 1.

2) With value 2 000, the ECU does not give information about which options of the standard are supported. These options concern use of normal or extended timing, additional length byte, header with or without address information.

In case of 5 Baud initialization the tester should know what options are implemented. In case of fast initialization the use of header and lenght byte will be the same as in the StartCommunicationSession positive response of the ECU.

5.2.4.2 Initialization with 5 Baud address word

5.2.4.2.1 CARB initialization

For CARB purposes 5 Baud initialization is used only. It is a functional initialization.Messages are send to all emission related ECUs (see ISO 9141-2 and ISO 14230-4).

5.2.4.2.2 Baud initialization

5.2.4.2.2.1 General

The general form of a 5 Baud initialization is shown in figure 10. The 5 Baud address byte is transferred from the tester on K-line and on L-line. After sending the 5 Baud address byte the tester will maintain L-line on high level.

After receiving the 5 Baud address byte the ECU will transmit the synchronization pattern "$55" and the two key bytes with the actual communication baud rate. The tester transmits key byte 2 (inverse), then the ECU transmits the address byte (inverse).

In the case of physical initialization, the ECU shall answer as shown in figure 10.

Using functional information (i.e. more than one ECU is initialized), the vehicle manufacturer shall take care that all ECUs use the same option of the protocol. Only one ECU shall perform the sequence of initialization (figure 10).

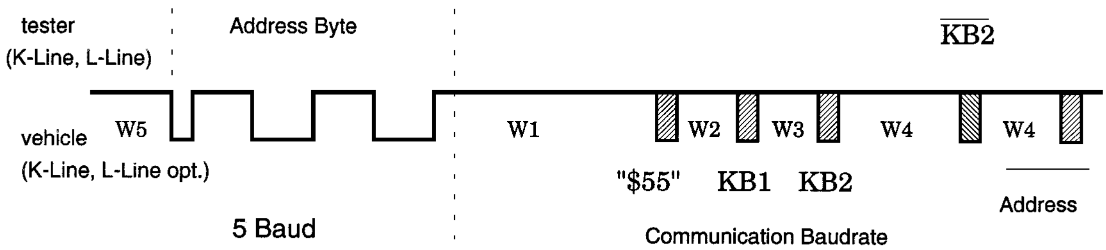


Figure 10 — 5 Baud initialization

Table 9 shows timing values for 5 Baud initialization. These are fixed values. They cannot be changed by the AccessCommunicationParameter service.

Table 9 — Timing values for 5 Baud initialization

Timing parameters	Values ms		Description
	min.	max.	
W1	60	300	Time from end of the address byte to start of synchronization pattern.
W2	5	20	Time from end of the synchronization pattern to the start of key byte 1.
W3	0	20	Time between key byte 1 and key byte 2.
W4	25	50	Time between key byte 2 (from the ECU) and its inversion from the tester. Also the time from the inverted key byte 2 from the tester and the inverted address from the ECU.
W5	300	—	Time before the tester starts to transmit the address byte.

5.2.4.2.2.2 Key bytes

Baud rates 1 200 to 10 400 Baud are allowed for communication. The tester will recognise the baud rate from the synchronization byte ($55).

5.2.4.2.2.3 Functional initialization

With this procedure a group of ECUs is initialized. Address bytes which define a functional group of ECUs are listed in annex A. Other manufacturer-defined functional address bytes in accordance with ISO 9141 (i.e. odd parity) are possible.

Functional addressing is only possible if all ECUs of a functional group use equal baud rates.

CARB-initialization is a special case of functional addressing.

5.2.4.2.2.4 Physical initialization

With this procedure, only a single ECU is initialized.

5-Baud-initialization address is as specified in ISO 9141. Odd parity is used. Address bytes are manufacturer-controlled.

5.2.4.2.3 Fast initialization

5.2.4.2.3.1 General

All ECUs which are initialized shall use a baud rate of 10 400 Baud for initialization and communication.

The tester transmits a Wake up Pattern (WuP) on K- and L-line synchronously. The pattern begins after an idle time on K-line with a low time of T_{iniL}. The tester transmits the first bit of the StartCommunication Service after a time of t_{Wup} following the first falling edge, as shown in figure 11. Values of T_{WuP} and T_{iniL} shall be as defined in table 10. There are different possibilities for the idle time T_{idle} :

— first transmission after power on : $T_{idle} \geqslant W_{5min}$;

— after completion of StopCommunication Service : $T_{idle} \geqslant P_{3min}$;

— after stopping communication by timeout P_{3max} : $T_{idle} \geqslant$ 0ms.

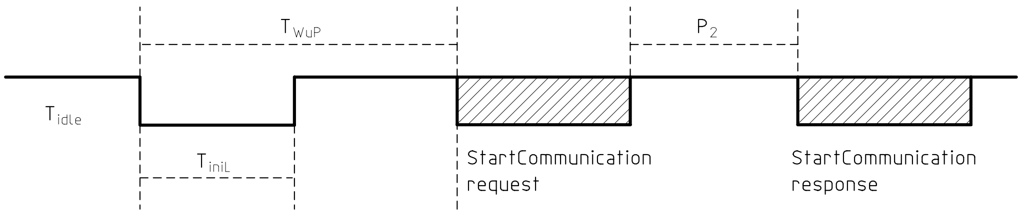


Figure 11 — Fast initialization

Table 10 — Timing values for fast initialization

Parameter		**min value**, ms	**max value**, ms
T_{iniL}	25 ± 1 ms	24 ms	26 ms
T_{WuP}	50 ± 1 ms	49 ms	51 ms

The transfer of a Wake up Pattern as described above is followed by a StartCommunicationRequest from the tester and a response from the ECU. The first message of a fast initialization always uses a header with target and source address and without additional length byte. An ECU may answer back with or without address information and length byte and tells its supported mode within the key bytes.

5.2.4.2.3.2 Bytes for messages

Tables 11 and 12 describe the different service StartCommunication messages.

Table 11 — StartCommunication Request Message

Byte No.	Parameter Name	CVT[1)]	Hex Value	Mnemonic
1	Format byte physical addressing fonctional addressing	M	$xx=[$81 $C1]	FMT
2	Target address byte	M	$xx	TGT
3	Source address byte	M	$xx	SRC
4	**StartCommunication Request Service Id**	**M**	**$81**	**SCR**
5	Checksum	M	$xx	CS
1) See 5.1.				

Table 12 — StartCommunication Positive Response Message

Byte No.	Parameter Name	CVT[1)]	Hex Value	Mnemonic
1	Format byte	M	$xx	FMT
2	Target address byte	C[2)]	$xx	TGT
3	Source address byte	C[2)]	$xx	SRC
4	Additional length byte	C[3)]	$xx	LEN
5	**StartCommunication Positive Response Service Id**	**S**	**$C1**	**SCRPR**
6	Key byte 1[4)]	M	$xx	KB1
7	Key byte 2[4)]	M	$xx	KB2
8	Checksum	M	$xx	CS
1) See 5.1. 2) Format byte is 10xx xxxx or 11xx xxxx. 3) Format byte is xx00 0000. 4) See 5.2.4.1 for use of key bytes.				

5.3 StopCommunication Service

5.3.1 Service definition

The purpose of this KWP 2000 communication layer service is to terminate a diagnostic communication.

5.3.2 Service table

See table 13.

Table 13 — StopCommunication Service

StopCommunication Request	M
StopCommunication Positive Response	S
StopCommunication Negative response	S
Response Code	M

5.3.3 Service Procedure

Upon receiving a StopCommunication indication primitive, the ECU shall check if the current conditions allow to terminate this communication. In this case, the Server shall perform all actions necessary to terminate this communication.

If it is possible to terminate the communication, the ECU shall issue a StopCommunication response primitive with the Positive Response parameters selected, before the communication is terminated.

If the communication cannot be terminated by any reason, the server shall issue an StopCommunication response primitive with the Negative Response parameter selected.

5.3.4 Byte implementation

Tables 14 to 16 describe the different StopCommunication request messages.

Table 14 — StopCommunication Request Message

Byte No.	Parameter name	CVT 1)	Hex Value	Mnemonic
1	Format byte	M	$xx	FMT
2	Target address byte	C 2)	$xx	TGT
3	Source address byte	C 2)	$xx	SRC
4	Additional length byte	C 3)	$xx	LEN
5	**StopCommunication Request Service Id**	**M**	**$82**	**SPR**
6	Checksum	M	$xx	CS
1) See 5.1 2) Format byte is 10xx xxxx or 11xx xxxx. 3) Format byte is xx00 0000.				

Table 15 — StopCommunication Positive Response Message

Byte No.	Parameter Name	CVT[1)]	Hex Value	Mnemonic
1	Format byte	M	$xx	FMT
2	Target address byte	C[2)]	$xx	TGT
3	Source address byte	C[2)]	$xx	SRC
4	Additional length byte	C[3)]	$xx	LEN
5	**StopCommunication Positive Response Service Id**	**S**	**$C2**	**SPRPR**
6	Checksum	M	$xx	CS

1) See 5.1.

2) Format byte is 10xx xxxx or 11xx xxxx.

3) Format byte is xx00 0000.

Table 16 — StopCommunication Negative Response Message

Byte No.	Parameter Name	CVT[1)]	Hex Value	Mnemonic
1	Format byte	M	$xx	FMT
2	Target address byte	C[2)]	$xx	TGT
3	Source address byte	C[2)]	$xx	SRC
4	Additional length byte	C[3)]	$xx	LEN
5	**Negative Response Service Id**	**S**	**$7F**	**SPRNR**
6	StopCommunication Request Service identification	**S**	$82	SCR
7	Response code[4)]= generalReject	M	$xx= $10	RC
8	Checksum	M	$xx	CS

1) See 5.1.

2) Format byte is 10xx xxxx or 11xx xxxx.

3) Format byte is xx00 0000.

4) Other response codes are possible: see ISO 14230-3.

5.4 AccessTimingParameter Service

5.4.1 Service definition

The purpose of this KWP 2000 communication layer service is to read and change the default timing parameters of a communication link for the communication link is active.

WARNING — Use of this service is complex; it depends on ECU capability and physical topology. The user of this service is responsible for its functionality.

5.4.2 Service table

See table 17.

Table 17 — AccessTimingParameter Service

a) AccessTimingParameter Request	
Timing Parameter Identifier (TPI)	M
P2min	C[1)]
P2max	C[1)]
P3min	C[1)]
P3max	C[1)]
P4min	C[1)]
b) AccessTimingParameter Positive Response	
Timing Parameter Identifier (TPI)	M
P2min	C[2)]
P2max	C[2)]
P3min	C[2)]
P3max	C[2)]
P4min	C[2)]
c) AccessTimingParameter Negative Response	**S**
Response Code	M
Timing Parameter Identifier (TPI)	M
1) Condition is TPI = Set values.	
2) Condition is TPI = Read limits, read current values.	

5.4.3 Service Procedure

5.4.3.1 This procedure has four different modes:

— read limits of possible timing parameters ;

— set timing parameters to default values ;

— read currently active timing parameters ;

— set timing parameters to given values.

5.4.3.2 Upon receiving an AccessTimingParameter indication primitive with TPI = 0, the ECU shall read the timing parameter limits, that is the values that the ECU is capable of supporting.

If the read access to the timing parameter is successful, the ECU shall send an AccessTimingParameter response primitive with the Positive Response parameters.

AccessTimingParameter response primitive with the Positive Response parameters.

If the read access to the timing parameters is not successfull, the ECU shall send an AccessTimingParameter response primitive with the Negative Response parameters.

5.4.3.3 Upon receiving an AccessTimingParameter indication primitive with TPI = 1, the server shall change all timing parameters to the default values and send an AccessTimingParameter response primitive with the Positive Response parameters before the defaults timing parameters become active.

If the timing parameters cannot be changed to default values for any reason, the ECU shall maintain the communication link and an AccessTimingParameter response primitive with the Negative Response parameters.

5.4.3.4 Upon receiving an AccessTimingParameter indication primitive with TPI = 2, the ECU shall read the currently used timing parameters.

If the read access to the timing parameters is successful, the ECU shall send an AccessTimingParameter response primitive with the Positive Response parameters.

If the read access no the currently used timing parameters is impossible for any reason, the ECU shall send an AccessTimingParameter response primitive with the Negative Response parameters.

5.4.3.5 Upon receiving an AccessTimingParameter indication primitive with TPI = 3, the ECU shall check if the timing parameters can be changed under the present conditions.

If the conditions are valid, the ECU shall perform all actions necessary to change the timing parameters and send an AccessTimingParameter response primitive with the Positive Response parameters before the new timing parameter limits become active.

If the timing parameters cannot be changed for any reason, the ECU shall maintain the communication link and send an AccessTimingParameter response primitive with the Negative Response parameters.

5.4.4 Implementation

Selection of mode (read/write/current/limits) is by the Timing Parameter Identifier (TPI), in accordance with table 18.

Table 18 — Selection of mode

Mode	TPI	CVT[1)]
Read limits	0000 0000B	C[2)]
Set parameters to default values	0000 0001B	-
Read current values	0000 0010B	C[3)]
Set values	0000 0011B	C[2)]

1) See 5.1.

2) Timing parameters are included in the request message if TPI = 3.

3) Timing parameters are included in the request message if TPI = 0 or 2.

5.4.5 Message bytes

Tables 19 to 21 describe the different service AccessTimingParameter messages.

Table 19 — AccessTimingParameter Request Message

Byte No.	Parameter Name	CVT[1)]	Hex Value	Mnemonic
1	Format byte	M	$xx	FMT
2	Target address byte	C[2)]	$xx	TGT
3	Source address byte	C[2)]	$xx	SRC
4	Additional length byte	C[3)]	$xx	LEN
5	**AccessTimingParameter Resquest Service Id**	**S**	**$83**	**ATP**
6	Timing Parameters Identifier = [read limits of poss. values, set parameter to default, read active parameters, set parameters]	M	$xx=[00,01, 02,03]	TPI
7	$P2_{min}$	C[4)]	$xx	$P2_{min}$
8	$P2_{max}$	C[4)]	$xx	$P2_{max}$
9	$P3_{min}$	C[4)]	$xx	$P3_{min}$
10	$P3_{max}$	C[4)]	$xx	$P3_{max}$
11	$P4_{min}$	C[4)]	$xx	$P4_{min}$
12	Checksum	M	$xx	CS

1) See 5.1.

2) Format byte is 10xx xxxx or 11xx xxxx.

3) Format byte is xx00 0000.

4) Timing parameters are included in the request message if TPI = 3.

Table 20 — AccessTimingParameters Positive Response Message

Byte No.	Parameter Name	CVT[1)]	Hex Value	Mnemonic
1	Format byte	M	$xx	FMT
2	Target address byte	C[2)]	$xx	TGT
3	Source address byte	C[2)]	$xx	SRC
4	Additional length byte	C[3)]	$xx	LEN
5	**AccessTimingParameters Positive Response Service Id**	**M**	**$C3**	**ATPPR**
6	Timing Parameter Identifier = [read of poss. values, set parameter to default, read active parameters, set parameters]	M	$xx = [00, 01, 02, 03]	TPI
7	$P2_{min}$	C[4)]	$xx	$P2_{min}$
8	$P2_{max}$	C[4)]	$xx	$P2_{max}$
9	$P3_{min}$	C[4)]	$xx	$P3_{min}$
10	$P3_{max}$	C[4)]	$xx	$P3_{max}$
11	$P4_{min}$	C[4)]	$xx	$P4_{min}$
12	Checksum	M	$xx	CS

1) See 5.1.

2) Format byte is 10xx xxxx or 11xx xxxx.

3) Format byte is xx00 0000.

4) Timing parameters are included in the request message if TPI = 0 or 2.

Table 21 — AccessTimingParameters Negative Response Message

Byte No.	Parameter name	CVT[1)]	Hex value	Mnemonic
1	Format byte	M	$xx	FMT
2	Target address byte	C[2)]	$xx	TGT
3	Source address byte	C[2)]	$xx	SRC
4	Additional length byte	C[3)]	$xx	LEN
5	**Negative Response Service Id**	**S**	**$7F**	**ATPNR**
6	AccessTimingParameters Request Service Id	M	$xx=$10	ATP
7	ResponseCode[4)]= generalReject	M	$xx=$10	RC
8	Checksum	M	$xx	CS

1) See 5.1.

2) Format byte is 10xx xxxx or 11xx xxxx.

3) Format byte is xx00 0000.

4) Other response codes are possible, see ISO 14230-3.

5.5 SendData Service

5.5.1 Service definition

The purpose of this KWP 2000 communication layer service is to transmit the data from the service resquest over a KWP2000 communication link.

5.5.2 Service table

See table 22.

Table 22 — SendData service

SendData Request	M
Service data	M
SendData Positive Response	S
SendData Negative Response	S
Response Code	M

5.5.3 Service Procedure

Upon a SendData request from the application layer, the respective data link layer entity of the message transmitter will perform all actions necessary to transmit the parameters of the request by a KWP 2000 message. This includes the determination of the message header (incl. the format byte), the concatenation of the message data, the checksum calculation, idle recognition, the transmission of message bytes and the timing surveillance (arbitration).

Upon receiving a message over a KWP 2000 communication link, the respective data link layer entity of the message receiver will perform all actions necessary to provide the received information to the respective application layer. This includes the recognition of a message start, the timing surveillance, the reception of message bytes, a checksum check, segmenting of the message data based on the format information and delivery of the message data to the application layer with a SendData indication primitive.

If the service was successfully complete (i.e. the message was transmitted), a SendData response primitive with the Positive Response parameter selected is delivered form the data link layer entity of the transmitting device to the respective application layer entity.

If the service cannot be performed by the data link layer entity of the transmission device, a SendData response primitive with the Negative Response parameters selected is delivered to the respective application layer entity.

6 Error handling

6.1 StartCommunication service

If the tester detects an error during the Start Communications Service either by timing or by the bit stream, then the tester will wait for a period of W_5 before beginning the process again (starting with the wake up pattern). If an ECU detects an error in the sequence from the tester then it shall be immediately prepared to recognise another StartCommunicationsService.

Bot tester and ECU are require to recognise failure to comply with maximum timing values. Minimum timing value transgressions need not be detected but are likely to cause bit stream errors.

6.2 Mainstream Communications

It is allowed but not required that the tester and ECUs may monitor their own messages. This creates four areas where error handling may be defined. Each is discussed separately below and diagrammatically represented in figure 12.

6.2.1 ECU-detected Tester transmission error

The ECU shall check each message by its checksum and number of bytes received before P2max. elapses. If either is in error then the ECU shall send no response and will internally ignore the whole message. The ECU is not required to check for other timing transgressions but may do so. Again no response should be given.

The ECU may detect other errors in the format or content of messages, but which satisfy the checksum and length requirements. In these cases, in order that the tester be aware that there is not a simple communications problem, the ECU should respond with the appropriate negative response message at least.

6.2.2 Tester-detected error in vehicle response

A single request may result in a single response from a single ECU or in multiple responses from several modules. The tester should check that all responses are correct both in terms of length and checksum as above. In case of an error or no response during P2max, it should retransmit the original message twice (i.e. three transmissions in total) before considering more severe error recovery processes. The application shall be informed of errors on the communication link. This will allow the application to make the appropriate action.

6.2.3 ECU-detected error in ECU response

The ECU may detect a difference between what it transmitted and what was detected on the K-line. In this case it may either do nothing or retry the transmission within a period of P_2 after bus activity ceasing. This allows various bus management methods to be adopted.

6.2.4 Tester-detected error in tester transmission

The tester may detect a difference between wat it transmitted and it detected on the K-line. In this case it shall transmit the whole message and give an error report to the application. After the time P2 max. the tester may retransmit the request.

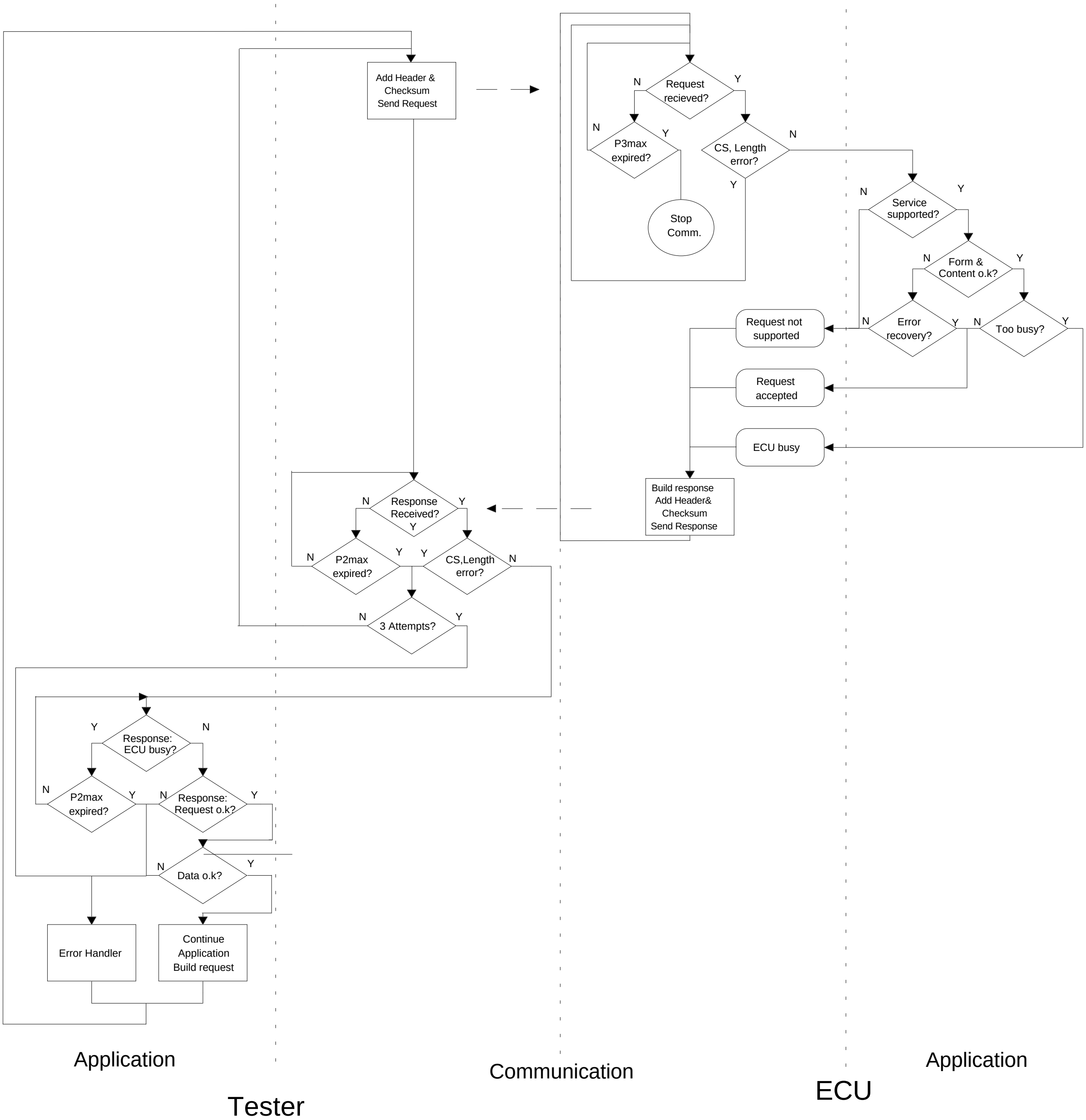


Figure 12 — Error handling

Annex A
(normative)

ECU/tester addresses for 5 Baud initialization

A.1 Physical addresses

Physical addresses shall be in accordance with ISO 9141.

The address byte consists of 1 start bit, 7 bit address, 1 parity bit (odd parity) and at least 1 stop bit.

Addresses are controlled by vehicle manufacturers.

A.2 Functional addresses

Addresses with values less than $80 are reserved for future standardization.

Addresses with values equal to or greater than $80 are manufacturer-specific.

Annex B
(informative)

ECU/tester addresses for fast initialization

Addresses for fast initialization may be the same as used for 5 Baud initialization (see annex A) or be in accordance with SAE J 2178, Part 1 (see table B.1).

Table B.1 — Addresses for fast initialization

Parameter	Address
Powertrain Controllers	
Integration/manufacturer Expansion	00 - 0F
Engine Controllers	10 - 17
Transmission Controllers	18 - 1F
Chassis Controllers	
Integration/Manufacturer Expansion	20 - 27
Brake controllers	28 - 2F
Steering Controllers	30 - 37
Suspension Controllers	38 - 3F
Body Controllers	
Integration/Manufacturer Expansion	40 - 57
Restraints	58 - 5F
Driver Information/Displays	60 - 6F
Lighting	70 - 7F
Entertainment/Audio	80 - 8F
Personal Communication	90 - 97
Climate Control (HVAC)	98 - 9F
Convenience (Doors, Seats, Windows, etc.)	C0 - C7
Security	C0 - C7
Future Expansion	C8 - CF
Manufacturer Specific	D0 - EF
Off-board Tester/Diagnostic Tools	F0 - FD
All Nodes	FE
Null Nodes	FF

Annex C
(informative)

Bibliography

[1] ISO 4092:1988, *Road vehicles — Diagnostic systems for motor vehicles — Vocabulary*.

[2] ISO 7637-1:1990, *Road vehicles — Electrical disturbance by conduction and coupling — Part 1: Passenger cars and light commercial vehicles with nominal 12 V supply voltage — Electrical transient conduction along supply lines only*.

[3] ISO/IEC 7498-1:1994, *Information technology — Open Systems Interconnection — Basic reference model: the basic model*.

[4] ISO 14229: — [1)], *Road vehicles — Diagnostic systems — Specification of diagnostic services*.

[5] ISO/IEC 10731:1994, *Information technology — Open Systems Interconnection — Basic reference model — Conventions for the definition of OSI services.*

[6] SAE J 2012:1996, *Recommended practice for diagnostic troubles codes definitions.*

[7] SAE J 2178/1:1995, *Class B Data Communication Network Messages — Part 1: Detailed header formats and physical address assignments*.

[1)] To be published.

ICS 43.180

Descriptors: road vehicles, motor vehicles, electronic equipment, electronic control units, diagnostic systems, digital technics, information interchange, protocols, data link layer.

Price based on 28 pages