SSF 14230 Road Vehicles - Diagnostic Systems

Keyword Protocol 2000 - Part 2 - Data Link Layer

Swedish Implementation Standard

Based on ISO 14230-2 Data Link Layer

Status: **Issue 1**

Date: **April 22, 1997**

This document is based on the International Standard ISO 14230 Keyword Protocol 2000 and has been further developed to meet Swedish automotive manufacturer's requirements by the Swedish Vehicle Diagnostics Task Force.
It is based on mutual agreement between the following companies:

- **Saab Automobile AB**
- **SCANIA AB**
- **Volvo Car Corp.**
- **Volvo Bus Corp.**
- **Mecel AB**

Document updates and issue history

This document can be revised and appear in several versions. The document will be classified in order to allow identification of updates and versions.

A. Document status classification

The document is assigned the status *Outline*, *Draft* or *Issue*.
It will have the *Outline* status during the initial phase when parts of the document are not yet written.
The *Draft* status is entered when a complete document is ready, which can be submitted for reviews. The draft is not approved. The draft status can appear between issues, and will in that case be indicated together with the new issue number E.g. *Draft Issue 2*.
An *Issue* is established when the document is reviewed, corrected and approved.

B. Version number and history procedure

Each issue is given a number and a date. A history record shall be kept over all issues.
Document in Outline and Draft status may also have a history record.

C. History

Issue #	Date	Comment
1	97 04 22	Frst issue

Table of Content

APPENDIX A - ARBITRATION

APPENDIX B - TIMING DIAGRAMS

APPENDIX C - MESSAGE FLOW EXAMPLES

Introduction

This document (The Swedish Keyword Protocol 2000 Implementation Standard) **is based on the ISO 14230-2 International Standard. Changes are indicated by changing the font from "Arial" to "Times New Roman"!**
It has been established in order to define common requirements for the implementation of diagnostic services for diagnostic systems.

To achieve this, the standard is based on the Open System Interconnection (O:S:I.) 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)

See figure 1 below.

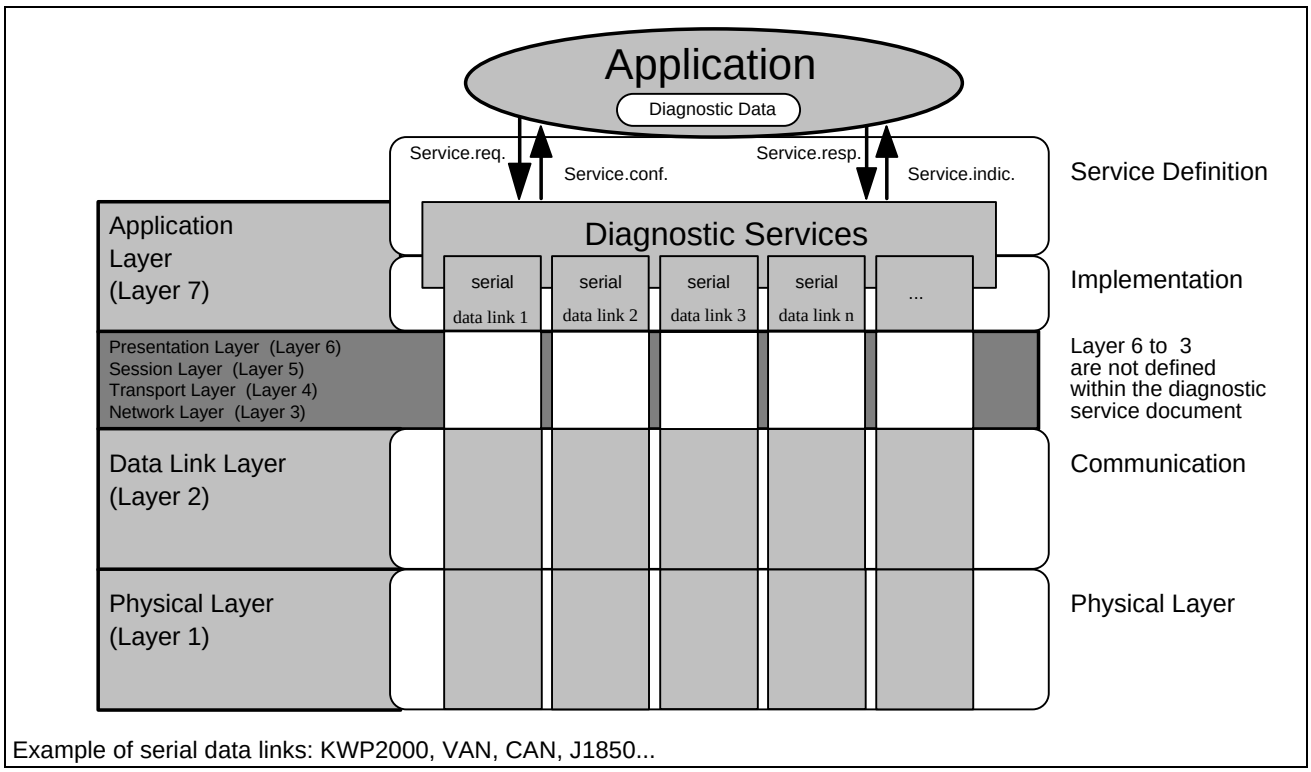


Figure 1 - Mapping of the diagnostic services on the OSI Model

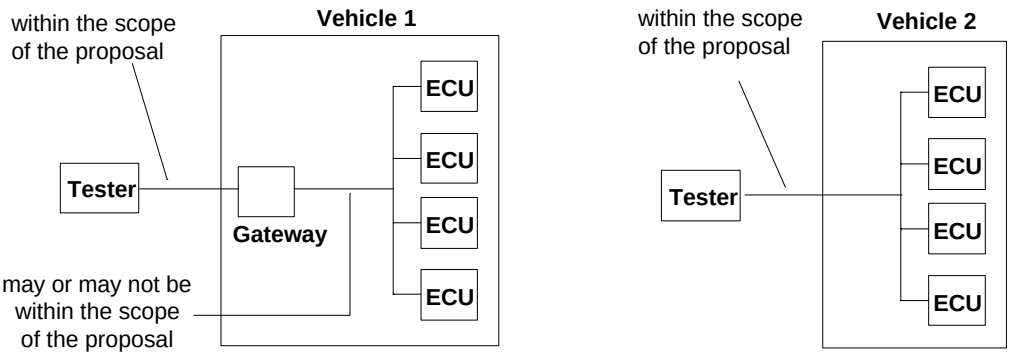
1. Scope

This national Standard specifies common requirements of diagnostic services which allow a tester to control diagnostic functions in an on-vehicle Electronic Control Unit (e.g. electronic fuel injection, automatic gear box, anti-lock braking system,...) 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 "Keyword Protocol 2000 - Part 3:Implementation) on a serial link (described in "Keyword Protocol 2000 - Part 1: Physical Layer") Also included are some communication services which are needed for communication/session management and a description of error handling.
This Standard does not specify the requirements for the implementation of diagnostic services.
The physical layer may be used as a multi-user-bus, so a kind of arbitration or bus management is necessary. If arbitration is used it shall comply to the technique described in Attachment A. The car manufacturers are responsible for the correct working of bus management.
Communication between ECUs are not part of this document.

The vehicle diagnostic architecture of this standard applies 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

See figure 2 below.



In vehicle 1, the ECUs are connected over an internal data link and indirectly connected to the diagnostic data link through a gateway. This document applies to the diagnostic communications over the diagnostic data link; the diagnostic communications over the internal data link may conform to this document or to another protocol.

In vehicle 2, the ECUs are directly connected to the diagnostic data link.

Figure 2 - Vehicle diagnostic architecture

2. Normative Reference

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

ISO 7498-1:1984	Information processing systems - Open systems interconnection - Basic reference model.
SAE J-1979:Dec,1991	E/E Diagnostic Test Modes
SAEJ-2178 :June, 1993	Class B Data Communication Network Messages
ISO 14229:1996	Road Vehicles - Diagnostic systems - Diagnostic Services Specification
SSF 14230-1:1997 Issue 2	Road Vehicles - Diagnostic systems - Keyword Protocol 2000 - Part 1: Physical Layer
SSF 14230-3:1996 Draft	Road Vehicles - Diagnostic systems - Keyword Protocol 2000 - Part 3: Implementation
ISO 14230-4:1996	Road Vehicles - Diagnostic systems - Keyword Protocol 2000 - Part 4: Requirements For Emission related Systems

3. Physical topology

Keyword Protocol 2000 is a bus concept (s. diagram below). Figure 3 shows the general form of this serial link.

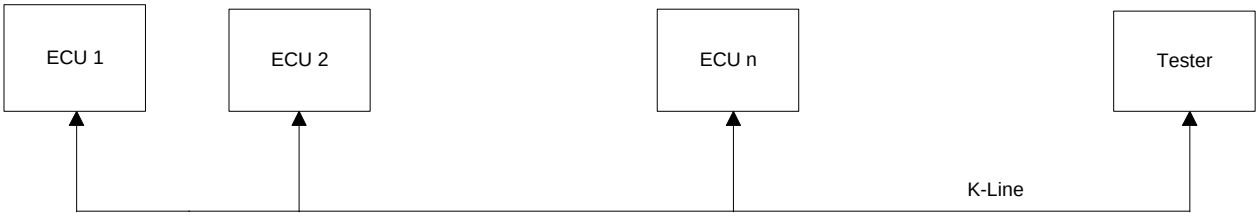


Figure 3 - Topology

The K-Line is used for communication and initialisation. Special cases are node-to-node-connections, that means there is only one ECU on the line, which also can be a bus converter.

4. Message structure

This section describes the structure of a message.
The message structure consists of three parts:

- header
- data bytes
- checksum

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

Header and Checksum byte are described in this document. The area of data bytes always begins with a Service Identification. Use of the data bytes for communication services is described in this document. Use of the data bytes for diagnostic services is described in "Keyword Protocol 2000 - Part 3: Implementation".

4.1 Header

The header consists of 3 or 4 bytes. A format byte includes information about the form of the message. A separate length byte allows message lengths up to 255 bytes.

4.1.1 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 the key bytes (s.5.1.2.1).

msb							lsb
A1	A0	L5	L4	L3	L2	L1	L0

- A1,A0: Define the form of the header which will be used by the message:

Header Mode	A1	A0	Mode	Mnemonic
2	1	0	Header with address information, physical target address	HM2
3	1	1	Header with address information, functional target address	HM3

HM0 and HM1 are not defined in this document.
HM3 (functional target address) shall only be used in request messages see §5.1.2.2.2

- L5..L0: Define the length of the data field of a message, i.e. from the beginning of the data field (Service Identification 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.
  In the Swedish Implementation Standard L0 to L5 shall always be set to 0 (except in the StartCommunicationRequest message, see §5.1.2.2).

4.1.2 Target address byte

This is the target address for the message. It may be a physical or a functional address. For emission related (CARB) messages this byte is defined in ISO 14230 KWP 2000 Part 4: Requirements For Emission related Systems.

4.1.2.1 Physical addressing

Physical addressing (HM2) can be used in both request and response messages. The target address of a physically addressed request shall be interpreted as a physical server (ECU) address, the source address is the physical address of the client (tester).
In the response message the target and source addresses are also physical addresses (HM2).
Physical addresses shall be according to SAE J2178-Part 1, or as specified by the vehicle manufacturer.

4.1.2.2 Functional addressing

Functional addressing (HM3) can only be used in request messages. The target address of a functionally addressed request shall be interpreted as a functional (group) address, the source address is the physical address of the client (tester).
In the response messages the target and source addresses are physical addresses, i.e. response messages are always physically addressed (HM2).
Functional addressing requires that the servers (ECUs) must support arbitration (see appendix A).

4.1.3 Source address byte

This is the address of the transmitting device. It must be a physical address (also in the case where the target address is a functional address). There are the same possibilities for the values as described for physical target address bytes. Addresses for testers are listed in SAE J2178 Part 1, but the ECU must accept all tester addresses.

4.1.4 Length byte

This byte is provided if the length in the header byte (L0 to L5) is set to 0. 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 the data field of a message, i.e. from the beginning of the data field (Service Identification byte included) to Checksum byte (not included). A data length of 1 to 255 bytes is possible. The longest message consists of a maximum of 260 byte (255 data bytes + 4 bytes header + Checksum). 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 may support both possibilities, the tester is informed about this capability through the keybytes (see section 5.1.2.1).

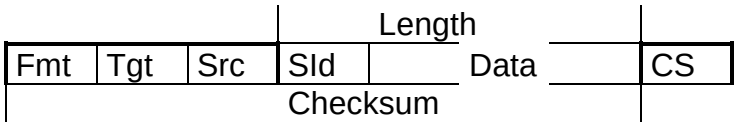
Length	Length provided in	
	Fmt byte	Length byte
< 64	XX00 0000	present
< 64	XXLL LLLL	not present
≥ 64	XX00 0000	present

XX: 2 bit address mode information (see §4.1.1)
LL LLLL: 6 bit length information

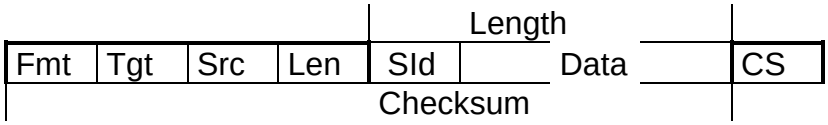
In the Swedish Implementation Standard the Length byte shall always be provided (L0 to L5 = 0) (except in the StartCommunicationRequest message, see §5.1.2.2).

4.1.5 Use of header bytes

With the above definitions there are two different forms of message. These are shown diagramatically below.



Header with address information, no additional length byte



Header with address information, with additional length byte

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

4.2 Data Bytes

The data field may contain up to 255 bytes of 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 "Keyword Protocol 2000 - Part 3: - Implementation" (for diagnostic services) and in section 5 of this document (for communication services).

4.3 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

<1> <2> <3> ... <N> , <CS>

where <i> ($1 \leq i \leq N$) is the numeric value of the i^{th} byte of the message, then:

$<CS> = <CS>_N$

where $<CS>_i$ (i = 2 to N) is defined as

$<CS>_i = \{ <CS>_{i-1} + <i> \}$ Modulo 256 and $<CS>_1 = <1>$

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

4.4 Timing

During normal operation the following timing parameters are relevant:

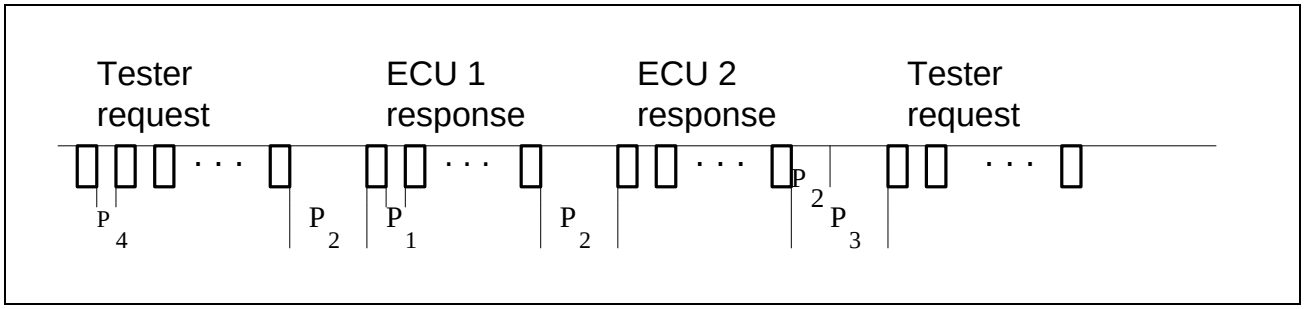


Figure 4 - Message flow, timing

Value	Description
P1	Inter byte time in ECU response.
P2	Time between end of tester request and start of ECU response, or time between end of ECU response and start of next ECU response. The next ECU response may be from the same ECU or it may be from another ECU in case of functional addressing.
P3	Time between end of ECU response and start of new tester request, or time between end of tester request and start of new tester request if ECU fails to respond. P3 shall be measured from the last byte in the latest response message from any ECU responding.
P4	Inter byte time in tester request.

There are two sets of default timing parameters, normal and extended. Only normal timing parameters are supported by this document (Swedish Implementation Standard).

Table 1a shows the timing parameters which are used as default (all values in ms).

Table 1a - Normal Timing Parameter Set, default values

Timing Parameter	min. values default	max. values default
P1	0	20
P2	25	50
P2*	25	5000
P3	55	5000
P4	5	20

Note: The timing parameter P2 becomes active if the server (ECU) responds with Negative response and the response code $78 "reqCorrectlyRcvd-RspPending", see §4.4.1.*

The values of the timing parameters may be changed with the communication service "AccessTimingParameters" (see §5.3).

Table 1b shows the resolution and the possible limits within which the timing parameters can be changed with AccessTimingParameters (ATP).

Table 1b - Normal Timing Parameter Set, lower and upper limits
All values in ms

Timing Parameter	Min. values		Max. values	
	Lower limit	Resolution [1]	Upper limit	Resolution [1]
P1	0	---	20	---
P2	0	0.5	89600 ; ∞	see Table 1c
P3	0	0.5	63500 ∞	250 see note [2]
P4	0	0.5	20	---

1) Min./Max. value calculation method [ms] = *ATP parameter value* * *Resolution*
2) *ATP parameter value* = $FF => Max. value = ∞

Table 1c - P2max Timing Parameter calculation

Timing Parameter	**Hex value** of *ATP parameter*	**Resolution** in [ms]	**value** in [ms]	**Maximum value calculation method** in [ms]
P2max	01 to F0	25	25 to 6000	(hex value) ✸ (Resolution)
	F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE	see maximum value calculation method	6400 12800 19200 25600 32000 38400 44800 51200 57600 64000 70400 76800 83200 89600	(low nibble of hex value) ✸ 256 ✸ 25 Example of $FA: ($0A ✸ $0100) ✸ 25 = 64000
	FF	---	∞	= ∞

The P2max 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 P2max timing values greater than 6000 [ms].

Calculation Formula for P2max values > $F0

Calculation_Of_P2max [ms] = (low nibble of ATP parameter P2max) * 256 * 25

Note: 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.

Users must take care for limits listed above and the following restrictions:

P3min > P2max (to avoid collisions in case of func. addressing or data segm.)
P3min > P4min (to guarantee that the ECU can receive the first byte)
Pimin < Pimax for i=1,...,4

When the tester and listening ECUs detect the end of a message by time-out, the following restrictions are also valid:

P2min > P4max
P2min > P1max

It is in the system designers responsibility to ensure proper communication in the case of changing the timing parameters from the default values.
He also has to make sure 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 possibly needs to leave its normal operation mode for switching over to a session with different communication parameters.

For complete timing diagrams see appendix B.

4.4.1 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 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.
After the transmission of the first negative response message, with response code $78, from the server (ECU) the timing parameter P2* becomes active, instead of the original timing parameter P2, in both the server and the client.
The timing parameter P2* shall be generated as described in the following formula:

P2*min = P2min
P2*max = P3max

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 (routine) initiated by the request message it shall send either a positive or 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 preceded by the negative response message(s) with response code $78, the timing parameter P2 becomes active again in both the server and the client. The client shall not repeat the request message after the reception of a negative response message with response code $78.

4.4.2 Periodic transmission

The Keyword Protocol 2000 Periodic Transmission Mode shall be enabled by starting a diagnostic session with the ***startDiagnosticSession*** service and the ***diagnosticMode (DCM_)*** parameter set to $82 for ***PeriodicTransmission***.
PeriodicTransmission shall be supported in connection with physical addressing, normal and modified timing. The description below explains in steps how the ***PeriodicTransmission*** mode shall be activated, handled and de-activated.

Step #1: To enable the ***PeriodicTransmission mode*** in the client (tester) and the server (ECU) the client (tester) shall transmit a ***startDiagnosticSession*** request message containing the ***diagnosticMode*** parameter for the ***PeriodicTransmission***. After the reception of the first positive response message from the server (ECU) the ***PeriodicTransmission*** mode is enabled and periodic transmission mode communication structure and timing becomes active. From now on, the server (ECU) shall periodically transmit the last response message with current (updated, if available) data content, until the client (tester) sends a request message within the timing window P3*. The timing parameters can be changed within the possible limits of the periodic transmission timing parameter set (see Table 1d) with the communication service AccessTimingParameters.

Step #2: After reception of any request message within the timing window P3*, the server (ECU) shall periodically transmit the corresponding response message which can be either a positive or a negative response message.

Step #3: After reception of a ***stopDiagnosticSession*** or ***stopCommunication*** request message within the timing window P3*, the server (ECU) shall transmit the corresponding positive response message only once.
After reception of a ***stopDiagnosticSession*** or ***stopComunication*** positive response message the ***periodicTransmissionMode*** is disabled and the default diagnostic session with the default timing values, defined by the key bytes becomes active.
After reception of a ***stopDiagnosticSession*** or ***stopComunication*** negative response message the ***periodicTransmissionMode*** shall continue. In such case the server (ECU) shall transmit negative response messages unless the client (tester) sends a new request message within the timing window P3*.

During the ***standardDiagnosticModeWithPeriodicTransmission*** the following rules have to be considered:

1. The client (tester) has to ignore the original timing window P3 and shall generate a new timing parameter for the jump-in timing window, which is called from now on P3*.
   The timing parameter P3* shall be generated as described in the following formula:
   P3*max = P2min - 2 ms
   P3*min = P3min
   Note: The original P3max timing parameter is only used for time out detection during negative response message handling with the response code $78 "reqCorrectlyRcvd-RspPending".

2. The timing window P3*, which starts at P3*min and ends at P3*max, shall be at least 5ms. It is important for the client (tester) to guarantee a minimum size of the jump-in window for the start of a request message.

3. The timing window P3* starts and ends **before** the timing window P2 starts.

4. P1max shall not exceed P2min. This is required in order to support resynchronisation between the server (ECU) and client (tester) to meet the error handling requirements.

5. Default and optimised timing parameter values
   The timing table below specifies the timing parameter values with the diagnostic mode ***standardDiagnosticModeWithPeriodicTransmission.***

Table 1d - Timing parameter - periodic transmission.
All values in ms

Timing Parameter	minimum values			maximum values		
	lower limit	default	resolution [1]	default	upper limit	resolution [1]
P1	0	**0**	---	**20**	20	0.5
P2	7	**25**	0.5	**50**	89600; ∞	see Table 1c
P2* [3]	7	**25**	0.5	**5000**	63500 ∞	250 see note [2]
P3	0	**5**	0.5	**5000**	63500 ∞	250 see note [2]
P3* [4]	0	**5**	0.5	**23**	125.5	0.5
P4	0	**5**	0.5	**20**	20	0.5

1) *Min./Max. value* calculation method [ms] = *ATP parameter value* * *Resolution*
2) *ATP parameter value* = $FF => *Max. value* = ∞
3) The timing parameter P2* becomes active if the server (ECU) responds with Negative response and the response code $78 "reqCorrectlyRcvd-RspPending", see §4.4.1
4) The timing parameter P3* can be changed, indirectly, by changing the timing parameters P2 and P3 with the service “AccessTimingParameters”.

6. When implementing the ***standardDiagnosticModeWithPeriodicTransmission*** the following limits and restrictions must be considered as listed below:
   - Pimin < Pimax for i=1, ,4
   - P1max < P2min
   - P3min ≤ P2min - 10ms

   It is the system designers responsibility to ensure proper communication in the case of changing the timing parameters from their default values.
   It is also the system designers responsibility to ensure proper communication when periodic transmission is used in combination with multiple diagnose, see §5.1.2.2.

For complete timing diagrams and message flow examples see appendix B and C.

4.4.3 Server (ECU) Response Data Segmentation

Server (ECU) Response Data Segmentation is used if a client (tester) has sent a request message which causes the server (ECU) to split the response message content (data bytes) into several data segments. The data segments shall be transmitted consecutively in repeated response messages. Each message shall be transmitted within the timing window P2. The data field of each response message shall consist of the Service ID and the corresponding data segment (see figure 5).
Data segmentation shall be detected by the client (tester) by comparing source addresses and Service IDs which must be identical for all response messages during segmentation.
Server (ECU) response data segmentation shall only be used when the data length exceeds the maximum length that the server (ECU) can transmit in a single message. Data segmentation shall not be supported in periodic transmission mode.
This procedure shall also be used to meet the requirements of ISO 14230-4 Keyword Protocol 2000 - Part 4: Requirements For Emission Related Systems.

If data segmentation is used the following restriction shall apply:

P3min > P2max

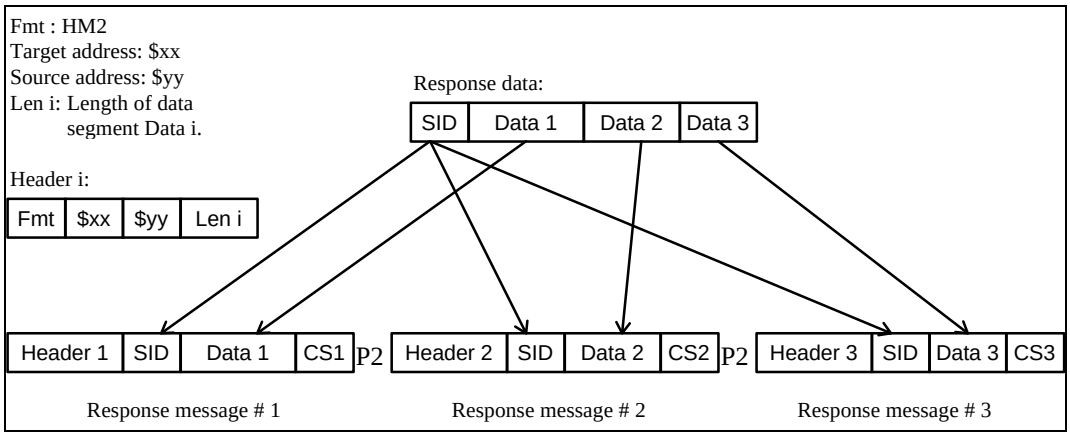


Figure 5 - Server (ECU) Response Data Segmentation

4.5 End Of Message

The end of a received message shall be detected as:

Number of bytes received equals message length (as defined in the format byte or length byte)

or

Time-out of inter byte time in the received message (P1max exceeded in ECU transmission, P4max exceeded in tester transmission)

whichever occurs first.

5. Communication services

Some services are necessary to establish and maintain communication. They are not diagnostic services because they do not appear on the application layer. They are described in the formal way and with the conventions defined in ISO/WD 14229, i.e. a definition of the service purpose, a service table and a verbal description of the service procedure. A description of implementation on the physical layer of Keyword Protocol 2000 is added.
The StartCommunication Service and the AccessTimingParameters Service are used for starting a diagnostic communication. In order to perform any diagnostic service, communication must be initialised and the communication parameters need to be appropriate to the desired diagnostic mode. A chart describing this is shown in figure 6.

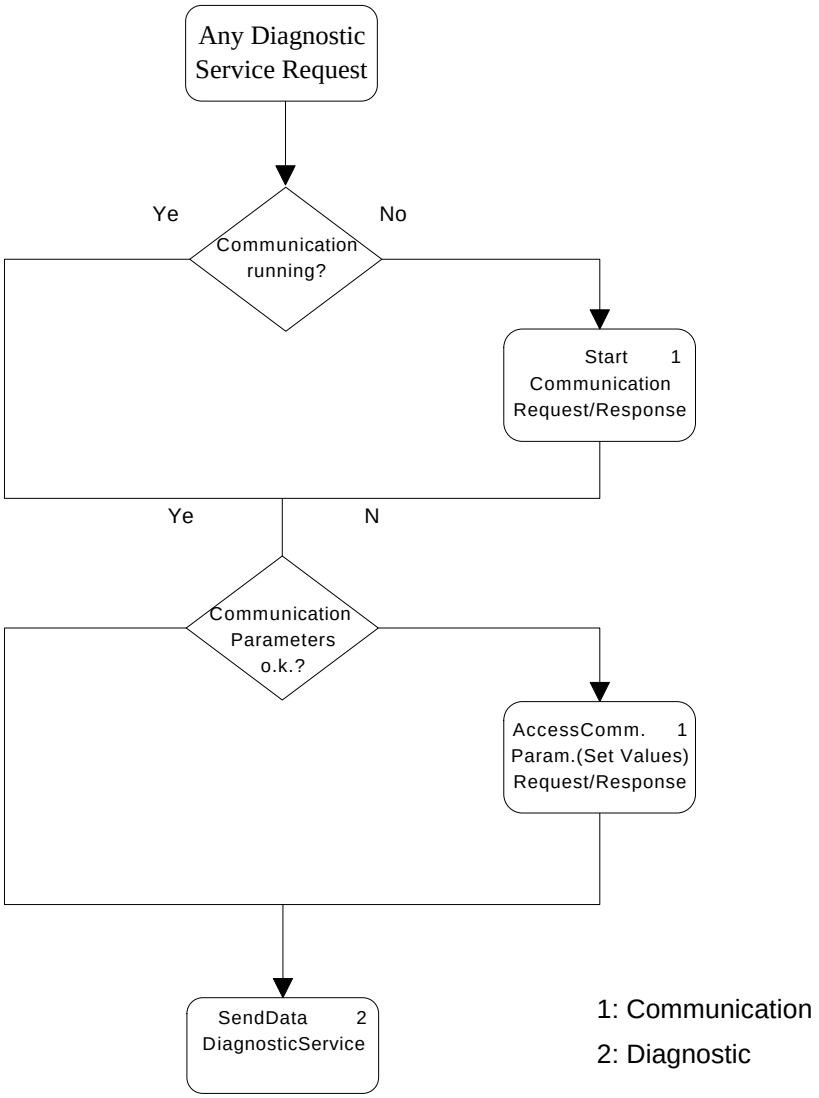


Figure 6 - Use of communication services

5.1 StartCommunication Service

5.1.1 Service Definition

5.1.1.1 Service Purpose

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

5.1.1.2 Service Table

Table 2 - StartCommunication Service

StartCommunication Request Target Initialisation Address Source Initialisation Address	M M M
StartCommunication Positive Response Keybytes Source Address	M M M

5.1.1.3 Service Procedure

Upon receiving a StartCommunication indication primitive, the ECU shall check if the requested communication link can be initialised under the present conditions. Valid conditions for the initialisation of a diagnostic communication link are described in section 5.1.2 "Implementation" of this document.
Then the ECU shall perform all actions necessary to initialise the communication link and send a StartCommunication response primitive with the Positive Response parameters selected.

If the communication link cannot be initialised by any reason, the ECU shall maintain its normal operation.

5.1.2 Implementation

The StartCommunication Service is used to initialise a communication on the K-line.
There are general facts that applies to the *fast initialisation* procedure:

- Prior to any activity there shall be a bus-idle time.
- Then the tester sends an initialisation pattern.
- All information which is necessary to establish communication is contained in the response of the ECU.

After finishing the initialisation the initialised ECUs are in the same status:

- 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 (= has a well defined functionality).

If an ECU that is already initialised (and has entered any diagnostic mode) receives a new StartCommunication Request (e.g. due to error recovery in the Tester) the request shall be accepted and the ECU shall be reinitialised.

5.1.2.1 Key bytes

With the key bytes an ECU informs the tester about the supported header, timing and length information.

The decoding of the key bytes is defined as:

KB1 (Low Byte) =:

Bit 0 (LSB) : AL0 (see table 3)
Bit 1 : AL1 (see table 3)
Bit 2 : HB0 (see table 3)
Bit 3 : HB1 (see table 3)
Bit 4 : TP0 (see table 3)
Bit 5 : TP1 (see table 3)
Bit 6: 1
Bit 7(MSB): Parity (odd)

KB2 (High Byte) =

Bit 0-6: $0F
Bit 7: 1 (odd parity)

Table 3 - Keybyte 1 compositions

	= 0	= 1
AL0	length inf. in format byte not supp.	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 addr. in header not supported	Tgt/Src address in header supported
TP0*	normal timing parameter set	extended timing parameter set
TP1*	extended timing parameter set	normal timing parameter set

* only TP0,TP1 = 0,1 and 1,0 allowed

The key bytes supported by this document (Swedish Implementation Standard) shall be:

KB1 $EA
KB2 $8F

i.e. Keyword 2026 ($8FEA)

The following table lists all possible key bytes. Shaded areas are not supported by this document:

Table 4 - Possible values of Keybytes

Keybytes Binary KB2	KB1	Hex	Dec.*	Supported Length information	Type of header	Timing
1000 1111	1101 0000	$8FD0	2000			
1000 1111	1101 0101	$8FD5	2005	format byte	1 Byte header	extended timing
1000 1111	1101 0110	$8FD6	2006	add. length byte		
1000 1111	0101 0111	$8F57	2007	both modes poss.		
1000 1111	1101 1001	$8FD9	2009	format byte	Header with target and source address information	
1000 1111	1101 1010	$8FDA	2010	add. length byte		
1000 1111	0101 1011	$8F5B	2011	both modes poss.		
1000 1111	0101 1101	$8F5D	2013	format byte	Both types of header supported	
1000 1111	0101 1110	$8F5E	2014	add. length byte		
1000 1111	1101 1111	$8FDF	2015	both modes poss.		
1000 1111	1110 0101	$8FE5	2021	format byte	1 Byte header	normal timing
1000 1111	1110 0110	$8FE6	2022	add. length byte		
1000 1111	0110 0111	$8F67	2023	both modes poss.		
1000 1111	1110 1001	$8FE9	2025	format byte	Header with target and source address information	
1000 1111	1110 1010	$8FEA	2026	add. length byte		
1000 1111	0110 1011	$8F6B	2027	both modes poss.		
1000 1111	0110 1101	$8F6D	2029	format byte	Both types of header supported	
1000 1111	0110 1110	$8F6E	2030	add. length byte		
1000 1111	1110 1111	$8FEF	2031	both modes poss.		

* Calculation of decimal value:
clear the parity bit of both keybytes
multiply keybyte 2 by 2^7 and add keybyte 1.

5.1.2.2 Fast Initialisation

All ECUs which are initialised must use a baud rate of 10400 baud for initialisation and communication.

The tester transmits a Wake up Pattern (WuP) on the K-Line. The pattern begins after an Idle time on K-line, T_{Idle}, 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 of the Wake up Pattern (see figure 8).

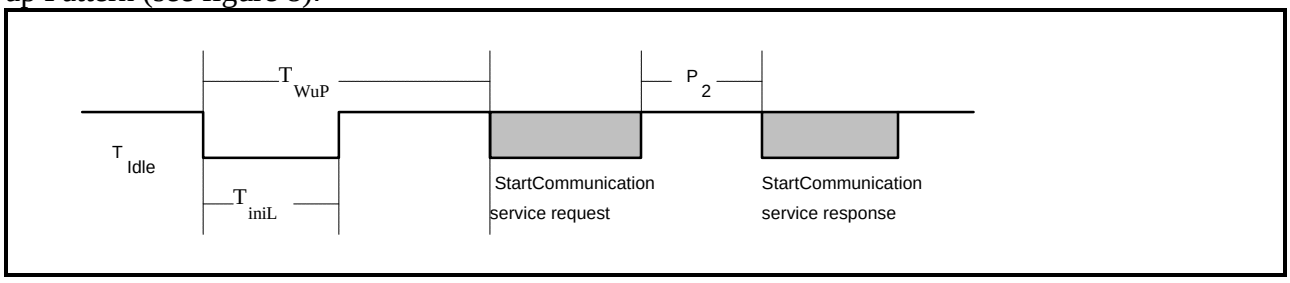


Figure 8 - Fast initialisation

Values of T_{WuP} and T_{iniL} are defined in table 5:

Table 5 - Timing values for fast initialisation

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

There are different possibilities for the Idle time T_{Idle}:

- First transmission after power on: $T_{Idle} \geq 300$ ms
- After completion of StopCommunication Service: $T_{Idle} \geq 55$ ms
- After stopping communication by time-out P3max: $T_{Idle} \geq 0$ ms

The transfer of a Wake up Pattern as described above is followed by a StartCommunication request from the tester and a response from the ECU. The first message of a fast initialisation always uses a header with target and source address and without additional length byte.

The StartCommunication Request message can be either physically or functionally addressed.

5.1.2.2.1 Physical initialisation

With this procedure a single server (ECU) is initialised. The format byte $81 (HM2) of the start communication request shall be used.

A physically addressed server (ECU), that can enter diagnose mode, shall answer back with a StartCommunication Positive Response.

The client (tester) shall have the possibility to make physically addressed initialisations of more than one server (ECU), sc. "multiple physical initialisations", by sending physically addressed StartCommunication Request messages to several servers (ECUs) (without sending StopCommunication in-between). In this case the Wake up Pattern shall be transmitted prior to each StartCommunication Request message. In this way a server (ECU) that is not yet initialised only has to listen for the Wake up Pattern. See appendix C - Message flow example.

This means that requests addressed to other initialised servers (ECUs) and responses addressed to the client (tester) from other initialised servers (ECUs) will appear on the K-line while the server (ECU) is initialised.

Reception of the Wake up Pattern, by servers (ECUs) already initialised, shall be handled with normal error handling (see § 6) and shall not result in a communication reset in the server (ECU).

Note: The timing requirements in §4.4 still prevents the client (tester) from having more than one active request at any one time.

5.1.2.2.2 Functional initialisation

With this procedure a group of servers (ECUs) are initialised. The format byte $C1 (HM3) of the start communication request shall be used. All addressed servers (ECUs), that can enter diagnose mode, shall answer back with a StartCommunication Positive Response.
During communication it is possible for the client (tester) to switch from functional to physical addressing (HM3 to HM2).

Functional initialisation requires that the servers (ECUs) must support arbitration (see appendix A).

StartCommunication Request Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$81 $C1]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$yy	SRC
#4	**startCommunication Request Service Id**	**M**	**$81**	**SCR**
#5	Checksum	M	$xx	CS

StartCommunication Positive Response Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$80 $C0]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$yy	SRC
#4	Additional length byte	M	$03	LEN
#5	**startCommunication Positive Response Service Id**	**S**	**$C1**	**SCRPR**
#6	Key byte 1	M	$EA	KB1
#7	Key byte 2	M	$8F	KB2
#8	Checksum	M	$xx	CS

5.2 StopCommunication Service

5.2.1 Service Definition

5.2.1.1 Service Purpose

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

5.2.1.2 Service Table

Table 6 - StopCommunication Service

StopCommunication Request Target Address	**M** M
StopCommunication Positive Response Source Address	**S** M
StopCommunication Negative Response Source Address Response Code	**S** M M

5.2.1.3 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.

If time-out of P3max is detected by the ECU, the communication shall be terminated without any response primitive being issued.

5.2.2 Implementation

StopCommunication Request Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$80 $C0]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$yy	SRC
#4	Additional length byte	M	$01	LEN
#5	**stopCommunication Request Service Id**	**M**	**$82**	**SPR**
#6	Checksum	M	$xx	CS

StopCommunication Positive Response Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$80 $C0]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$yy	SRC
#4	Additional length byte	M	$01	LEN
#5	**stopCommunication Positive Response Service Id**	**S**	**$C2**	**SPRPR**
#6	Checksum	M	$xx	CS

StopCommunication Negative Response Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$80 $C0]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$yy	SRC
#4	Additional length byte	M	$03	LEN
#5	**negative Response Service Id**	**S**	**$7F**	**SPRNR**
#6	stopCommunication Request Service Identification	M	$82	SCR
#7	ResponseCode* = generalReject	M	$xx=$10	RC
#8	Checksum	M	$xx	CS

* Other response codes possible, see Keyword Protocol 2000 - Part 3: Implementation

5.3 AccessTimingParameter Service

5.3.1 Service Definition

5.3.1.1 Service Purpose

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

5.3.1.2 Service Table

Table 7 - AccessTimingParameter Service

AccessTimingParameter Request	**S**
Target Address	M
Timing Parameter Identifier (TPI)	M
P2min	C1
P2max	C1
P3min	C1
P3max	C1
P4min	C1
AccessTimingParameter Positive Response	**S**
Source Address	M
Timing Parameter Identifier (TPI)	M
P2min	C2
P2max	C2
P3min	C2
P3max	C2
P4min	C2
AccessTimingParameter Negative Response	**S**
Source Address	M
Response Code	M
Timing Parameter Identifier (TPI)	M

C1: Condition is TPI = Set values

C2: Condition is TPI = Read limits, read current values

5.3.1.3 Service Procedure

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

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

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 default 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 send an AccessTimingParameter response primitive with the Negative Response parameters.

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 to the currently used timing parameters is impossible for any reason, the ECU shall send an AccessTimingParameter response primitive with the Negative Response parameters.

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 by any reason, the ECU shall maintain the communication link and send an AccessTimingParameter response primitive with the Negative Response parameters.

5.3.2 Implementation

Selection of mode (read/write/current/limits) is by the Timing Parameter Identifier (TPI):

MODE		TPI	Cond.
Read	limits	0000 0000B	C2
Set	parameters to default values	0000 0001B	-
Read	current values	0000 0010B	C2
Set	values	0000 0011B	C1

AccessTimingParameter Request Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$80 $C0]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$xx	SRC
#4	Additional length byte	M	$xx	LEN
#5	**AccessTimingParameters Request Service Id**	**M**	**$83**	**ATP**
#6	Timing Parameter Identifier =[read limits of poss.values, set parameter to default, read current values, set parameters]	M	$xx=[00, 01, 02, 03]	TPI
#7	P2min	C1	$xx *	P2MIN
#8	P2max	C1	:	P2MAX
#9	P3min	C1	:	P3MIN
#10	P3max	C1	:	P3MAX
#11	P4min	C1	$xx	P4MIN
#12	Checksum	M	$xx	CS

AccessTimingParameter Positive Response Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$80 $C0]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$xx	SRC
#4	Additional length byte	M	$xx	LEN
#5	**AccessTimingParameters Positive Response Service Id**	**S**	**$C3**	**ATPPR**
#6	Timing Parameter Identifier =[read limits of poss.values, set parameter to default, read current values, set parameters]	M	$xx=[00, 01, 02, 03]	TPI
#7	P2min	C2	$xx *	P2MIN
#8	P2max	C2	:	P2MAX
#9	P3min	C2	:	P3MIN
#10	P3max	C2	:	P3MAX
#11	P4min	C2	$xx	P4MIN
#12	Checksum	M	$xx	CS

AccessTimingParameters Negative Response Message

Byte #	Parameter Name	CVT	Hex Value	Mnemonic
#1	Format byte physical addressing functional addressing	M	$xx=[$80 $C0]	FMT
#2	Target address byte	M	$xx	TGT
#3	Source address byte	M	$xx	SRC
#4	Additional length byte	M	$03	LEN
#5	**Negative Response Service Id**	**S**	**$7F**	**ATPNR**
#6	**AccessTimingParameters** Request Service Identification	M	$83	ATR
#7	ResponseCode** = generalReject	M	$xx=$10	RC
#8	Checksum	M	$xx	CS

* The values of the timing parameters shall be calculated as: $xx = time (in ms) / resolution (as spec. in table 1)

** Other response codes possible, see Keyword Protocol 2000 - Part 3: Implementation

C1: TPI = Set values

C2: TPI = Read limits, read current values

5.4 SendData Service

5.4.1 Service Definition

5.4.1.1 Service Purpose

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

5.4.1.2 Service Table

Table 8 - SendData Service

SendData Request	M
Target Address	M
Service Data	M
SendData Positive Response	**S**
Source Address	M
Service Data	M
SendData Negative Response	**S**
Source Address	M
Response Code	M

5.4.1.3 Service Procedure

Upon a SendData request/response 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/response by a KWP2000 message. This includes the determination of the message header (incl. the format byte and source address), 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 KWP2000 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 (incl. the recognition of the format byte and target address), 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/confirmation primitive.

5.4.2 Implementation

The implementation of different diagnostic services is defined in Keyword Protocol 2000 - Part 3: Implementation.

In order to avoid consecutive time-outs of P3max in the ECU the tester must issue a new request within the range of P3min - P3max after the latest response from an ECU (except in *periodicTransmissionMode*).
If no other service is requested by the user the tester shall use the "Tester Present" service as defined in Part 3: Implementation.

6. Error Handling

6.1 Error handling during physical/functional Fast Initialisation

6.1.1 Client (tester) Error Handling during physical/functional Fast Initialisation

Client (tester) detects an...	Action
... error in T_{idle} (W5 or P3min)	The client (tester) is responsible to keep to the idle time. The server (ECU) is responsible to keep to the time P1min. In case of an error the client (tester) must wait for T_{idle} again.
... error in P1min	No observation necessary. P1min is always 0 ms.
... error in P1max (P1max time-out)	The client (tester) shall ignore the response and shall open a new timing window P2 to receive a directly repeated response from the same server ECU) or a response from another server (ECU). If the server (ECU) does not repeat the response, the client (tester) shall wait for P3min and afterwards the client (tester) may start a new initialisation beginning with a wake up pattern.
... error in P2min	No observation necessary. P2min is always 0 ms during initialisation.
... error in P2max (no valid response from any server (ECU))	If the client (tester) does not receive any response, the client (tester) shall wait for P3min and afterwards the client (tester) may start a new initialisation beginning with a wake up pattern.
... error in StartComunication Positive Response (Byte collision) (Response contents) (Response checksum)	The client (tester) shall ignore the response and shall open a new timing window P2 to receive a directly repeated response from the same server (ECU) or a response from another server (ECU). If the server (ECU) does not repeat the response, the client (tester) shall wait for P3min and afterwards the client (tester) may start a new initialisation beginning with a wake up pattern.

Client (tester) Error Handling during physical/functional Fast Initialisation

6.1.2 Server (ECU) Error Handling during physical Fast Initialisation

Server (ECU) detects an ...	Action
... error in T_{idle} (W5 or P3min)	No observation necessary. The client (tester) is responsible to keep to the idle time T_{idle} .
... error in wake-up-pattern	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.
... error in P4min	No observation necessary. The client (tester) is responsible to keep to the time P4min.
... error in P4max (P4max time-out)	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.
... error in StartComunication Request (checksum, contents)	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.
... not allowed client (tester) source address or server (ECU) target address	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.

Server (ECU) Error Handling during physical Initialisation

6.1.3 Server (ECU) Error Handling during functional Fast Initialisation

Server (ECU) detects an ...	Action
... error in T_{idle} (W5 or P3min)	No observation necessary. The client (tester) is responsible to keep to the idle time T_{idle}.
... error in wake-up-pattern	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.
... error in P4min	No observation necessary. The client (tester) is responsible to keep to the time P4min
... error in P4max (P4max time-out)	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.
... error in StartComunication Request (checksum) (contents)	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.
... error in StartComunication Positive Response (Byte collision)	The server (ECU) shall repeat the response within a new timing window P2 considering arbitration.
... not allowed client (tester) source address or server (ECU) target address	The server (ECU) shall not respond and shall be able to detect immediately a new wake-up-pattern sequence.

Server (ECU) Error Handling during functional Initialisation

6.2 Error handling after Initialisation

6.2.1 Client (tester) communication Error Handling

Client (tester) detects an ...	Action
... error in P1min	No observation necessary. P1min is always 0 ms.
... error in P1max (P1max time-out)	The client (tester) shall ignore the response and shall open a new timing window P2 to receive a directly repeated response from the same server (ECU) or a response from another server (ECU). If the server (ECU) does not repeat the response, the client (tester) shall repeat the same request in a new timing window P3. This shall be done twice if necessary (i.e. three transmission in total).
... error in P2min	No observation necessary. The server (ECU) is responsible to keep to the time P2min.
... error in P2max (no valid response from any server (ECU) or missing responses)	The client (tester) shall repeat the last request in a new timing window P3 This shall be done twice if necessary (i.e. three transmission in total). Any following appropriate action is client (tester) dependent if the client (tester) does not receive a response.
... error in server (ECU) Response (checksum) (contents)	The client (tester) shall repeat the last request in a new timing window P3 This shall be done twice if necessary (i.e. three transmission in total). Any following appropriate action is client (tester) dependent if the client (tester) does not receive a response. If multiple responses are received with errors, any correct responses shall be presented to the application.

Client (tester) communication Error Handling after physical/functional Initialisation

6.2.2 Server (ECU) communication Error Handling. physical addressing

Server (ECU) detects an ...	Action
... error in P3min	No observation necessary. If a request is received it shall be ignored. The client (tester) is responsible to keep to the time P3min.
... error in P3max (P3max time-out)	The server (ECU) shall reset communication and shall be able to detect immediately a new wake-up-pattern sequence.
... error in P4min	No observation necessary The client (tester) is responsible to keep to the time P4min
... error in P4max (P4max time-out)	The server (ECU) shall ignore the request and shall open a new timing window P3 to receive a new request from the client (tester).
... error in client (tester) Request (header or checksum)	The server (ECU) shall ignore the request and shall open a new timing window P3 to receive a new request from the client (tester).
... error in client (tester) Request (contents) (not SendData service)	In order that the client (tester) be aware that there is not a simple communications problem, the server (ECU) shall respond with the appropriate negative response message.
...not allowed source or target address	The server (ECU) shall ignore the request and shall open a new timing window P3 to receive a new request from the client (tester).
...error in its own Response	If the server (ECU) sends repeated responses it must detect if the client (tester) is no longer present, i.e. no pull-up on the K-line. The server shall then return to normal operation mode.

Server (ECU) communication Error Handling after physical Initialisation

6.2.3 Server (ECU) Error Handling, functional addressing

Server (ECU) detects an ...	Action
... error in P3min	No observation necessary. The client (tester) is responsible to keep to the time P3min.
... error in P3max (P3max time-out)	The server (ECU) shall reset communication and shall be able to detect immediately a new wake-up-pattern sequence.
... error in P4min	No observation necessary. The client (tester) is responsible to keep to the time P4min.
... error in P4max (P4max time-out)	The server (ECU) shall ignore the request and shall open a new timing window P3 to receive a new request from the client (tester).
... error in client (tester) request (header or checksum)	The server (ECU) shall ignore the request and shall open a new timing window P3 to receive a new request from the client (tester).
... error in client (tester) Request (contents) (not SendData service)	In order that the client (tester) be aware that there is not a simple communications problem, the server (ECU) shall respond with the appropriate negative response message.
...not allowed source or target address	The server (ECU) shall ignore the request and shall open a new timing window P3 to receive a new request from the client (tester).
... error in its own Response (byte collision)	If the server (ECU) detects a byte collision within its own response, it must repeat the response within a new timing window P2 considering the arbitration.

Server (ECU) Error Handling after functional Initialisation

Appendix A - Arbitration

1. Definitions

When more than one ECU can respond to a single request there exists the possibility of collisions between competing response messages.
This appendix describes the method to be used for detecting collisions and modifying response timing to avert further collisions.
This is applicable to functionally addressed requests following fast initialisation.
The method described complies to the official ISO proposal made by Lucas/Bosch.

Arbitration is composed of the following elements:

1.1 Random response time

After each received request the ECU shall calculate a random response time. The diagnose address shall be used as seed for the algorithm used for calculating random values.

1.2 Start bit detection

This is used by the ECU to detect the start of transmission of a response message from another ECU. It indicates the detection of the falling edge of the first start bit.
When a start bit is detected the ECU may not transmit its own response message until the ongoing message is completed.

1.3 Transmission latency

If no start bit is detected the bus is defined as idle and the ECU may transmit its own response message. The latency between bus idle detection and start of transmission (transmission latency) must not be longer than 0.1 ms.

1.4 Collision detection

The ECU must, for each byte it transmits, detect a (possible) difference between what it transmitted and what it detected on the bus.

2. Mainstream Communication

Arbitration, as defined above, shall be used during communication of response messages to functionally addressed requests following fast initialisation (including StartCommunicationRequest).

When a valid request has been received the ECU shall enable start bit detection and calculate a random response time, $P2_{random}$.
The start bit detection shall be enabled within the first 80% of the time period between the end of the tester request and P2min.
$P2_{random}$ shall be within the range of time period P2 (P2min - P2max), where at least 80% of the range shall be used for the calculated random value. The maximum resolution between the calculated random response times shall be 1 ms.
If no start bit has been detected by $P2_{random}$ the response message shall be transmitted.

If the ECU looses arbitration, i.e. a start bit is detected before $P2_{random}$, or if a collision occurs, i.e. the ECU detects a difference between what it transmitted and what it detected on the bus, the ECU shall abort the transmission and wait for the end of the ongoing transmission of a message. When the end of message is detected (either as a valid or a corrupted message) the ECU shall again enable start bit detection and calculate a new $P2_{random}$. If no start bit has been detected by the new $P2_{random}$ the response message shall be retransmitted.

Appendix B - Timing diagrams

1. Physical addressing

1.1 Physical addressing - single positive response message

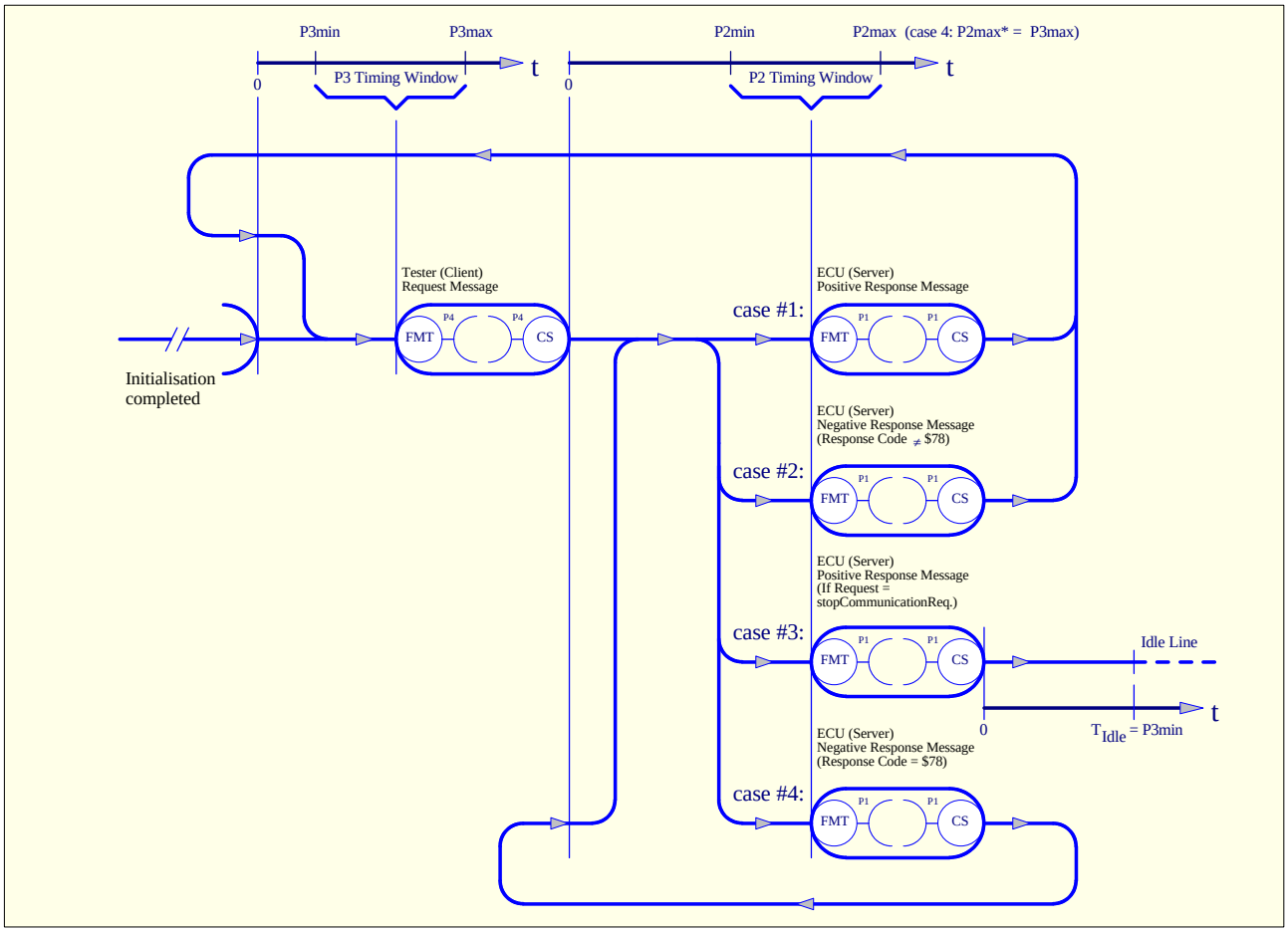


Figure B1.1 - Physical addressing - single positive response message

Above figure B1.1 is based on a physical addressing (fast initialisation) followed by a client (tester) request message (target address = single server (ECU)) and a single response message from the server (ECU). This supports four different cases (see table below).

Table B1.1 - Physical addressing - single positive response message

Case	Description
#1	This case specifies a **single positive response message (not stopCommunication)** of the server (ECU) preceded by a request message (not stopCommunication) of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#2	This case specifies a **single negative response message with the negative response code NOT equal to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#3	This case specifies a **single positive StopCommunication response message** of the server (ECU) preceded by a stopCommunication request message of the client (tester). The response message is sent within the P2 timing window. **The P3 time, which has been started after completion of the stopCommunication response message, must first reach the active P3min = T_{Idle} timing value before an "Idle Line" becomes active.**

#4	This case specifies one or multiple **negative response message with the negative response code set to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). **The first response message is sent within the P2 timing window.** Response code '$78' causes the server (ECU) and the client (tester) to modify the P2max timing parameter to change to the P2max*=P3max timing parameter. The client (tester) shall NOT send any request message. This shall provide the server (ECU) more time to prepare for the succeeding response message. The negative response message is followed by another response message as described in case #1, #2. #3 and #4. It depends on the request message of the client (tester) and/or the behaviour of the server (ECU) whether case #1, #2, #3 or #4 becomes active. After completion of case #1, #2 or #3 the P2max*=P3max timing parameter is reset (in both, server (ECU) and client (tester)) to the previous P2max timing parameter after successful completion of the response message. The timing conditions are kept as long as case #4 is repeated. **Note: Case #4 must always be terminated with case #1 or #2 or #3 (only if request message = stopCommunication)!**

1.2 Physical addressing - more than one positive response message

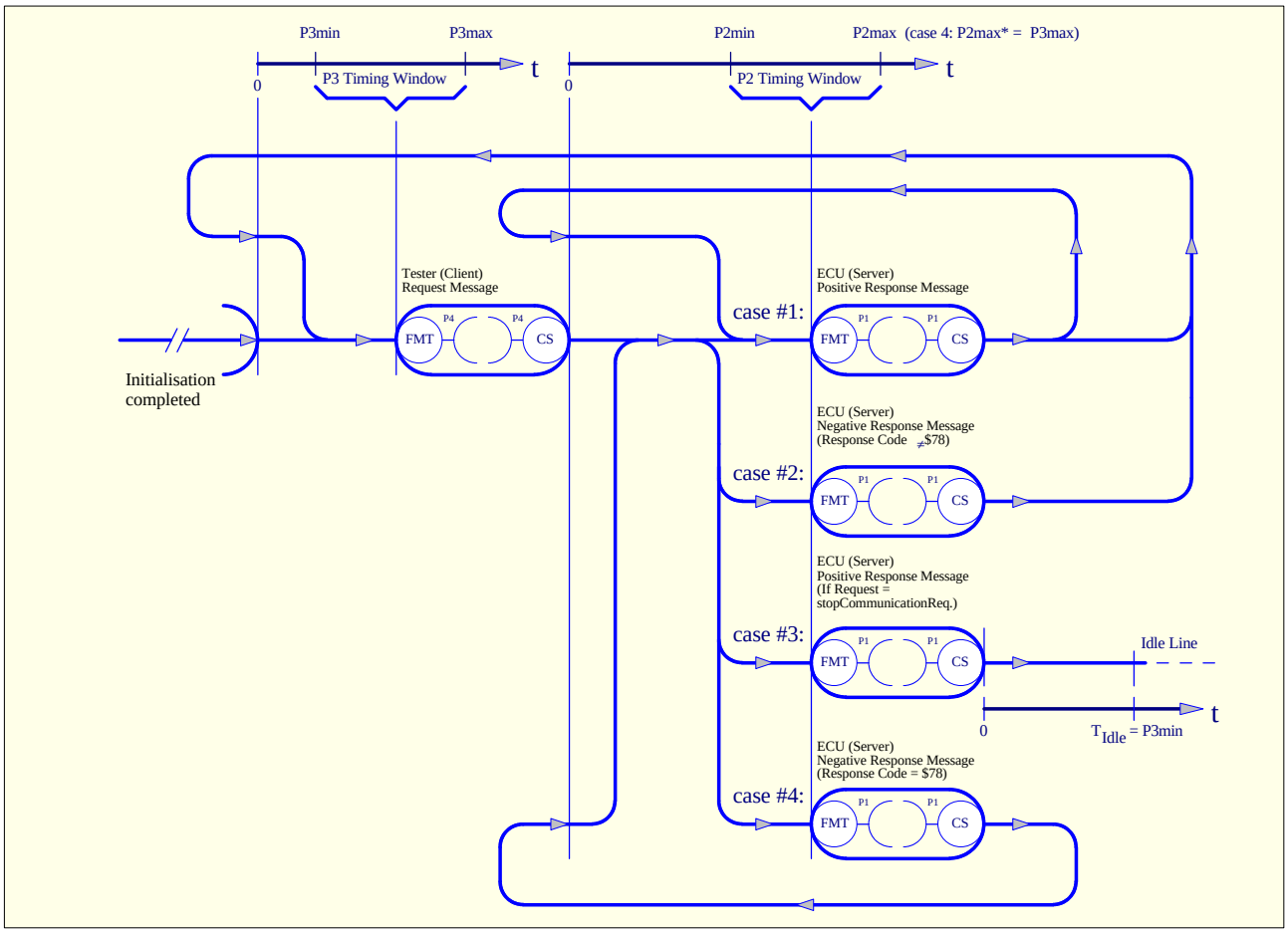


Figure B1.2 - Physical addressing - more than one positive response message

Above figure B1.2 is based on a physical addressing (fast initialisation) followed by a client (tester) request message (target address = single server (ECU)) and more than one positive response message from the server (ECU). This supports four different cases (see table below).

Table B1.2 - Physical addressing - more than one positive response message

Case	Description
#1	This case specifies **more than one positive response messages (not stopCommunication)** of the server (ECU) preceded by a request message (not stopCommunication) of the client (tester). **The response messages are sent within the P2 timing window. After completion of all positive response messages the last is followed by a client (tester) request message within the P3 timing window.** **Note:** More than one positive response message requires data byte accumulation handling in the client (tester). After a positive response message only further positive response messages are allowed (no negative response messages).
#2	This case specifies a **single negative response message with the negative response code NOT equal to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#3	This case specifies a **single positive stopCommunication response message** of the server (ECU) preceded by a stopCommunication request message of the client (tester). The response message is sent within the P2 timing window. **The P3 time, which has been started after completion of the stopCommunication response message, must first reach the active P3min = T_{Idle} timing value before an "Idle Line" becomes active.**

#4	This case specifies one or multiple **negative response message with the negative response code set to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). **The first response message is sent within the P2 timing window.** Response code '$78' causes the server (ECU) and the client (tester) to modify the P2max timing parameter to change to the P2max*=P3max timing parameter. The client (tester) shall NOT send any request message. This shall provide the server (ECU) more time to prepare for the succeeding response message. The negative response message is followed by another response message as described in case #1, #2. #3 and #4. It depends on the request message of the client (tester) and/or the behaviour of the server (ECU) whether case #1, #2, #3 or #4 becomes active. After completion of case #1, #2 or #3 the P2max*=P3max timing parameter is reset (in both, server (ECU) and client (tester)) to the previous P2max timing parameter after successful completion of the response message. The timing conditions are kept as long as case #4 is repeated. **Note: Case #4 must always be terminated with case #1 or #2 or #3 (only if request message = stopCommunication)!**

1.3 Physical addressing - periodic transmission

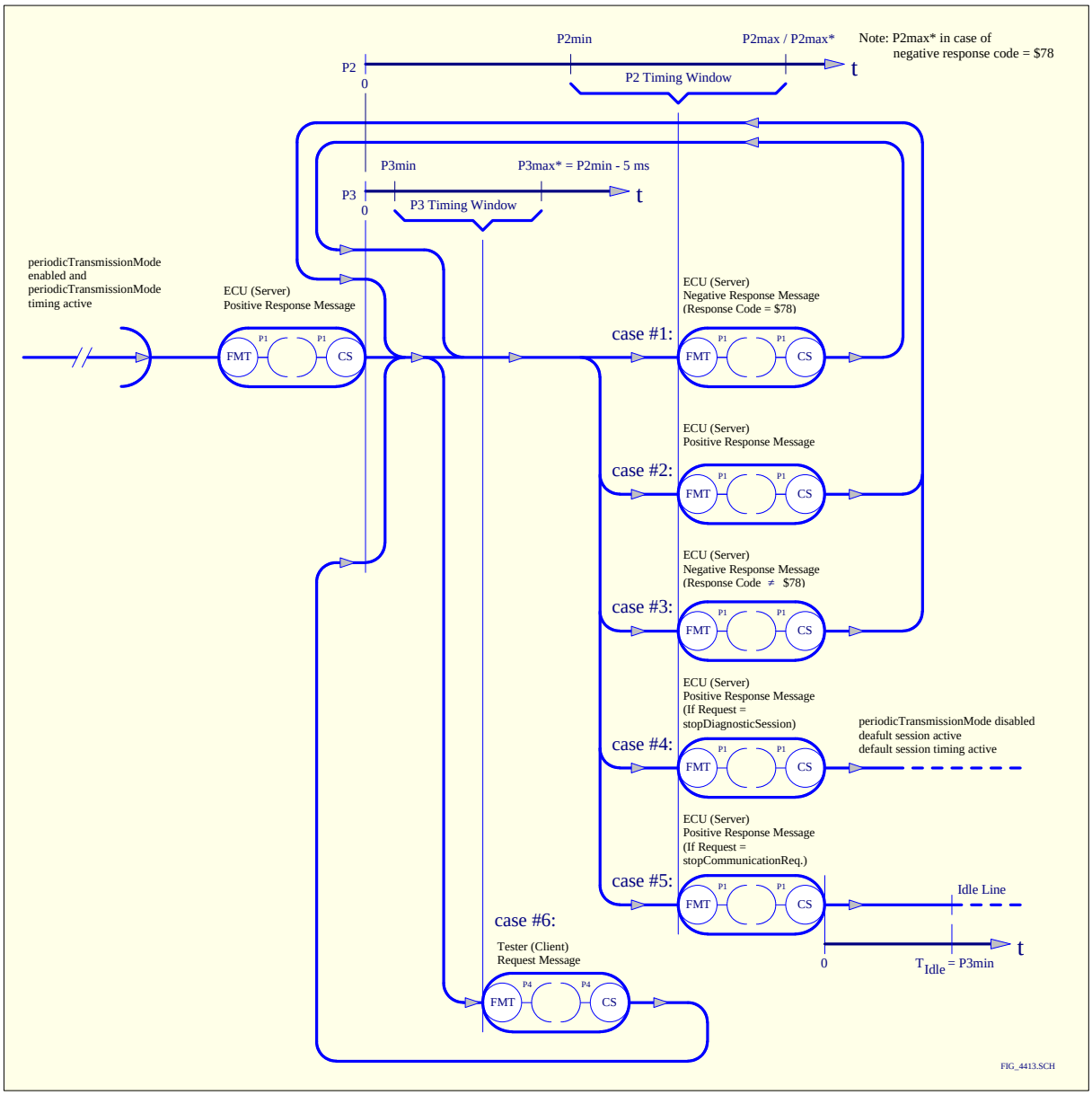


Figure B1.3- Physical addressing - periodic transmission

Above figure B1.3 is based on a physical addressing (fast initialisation) with the diagnostic mode ***PeriodicTransmission*** activated. The client (tester) has transmitted a request message (target address = single server (ECU)) which is followed by periodically transmitted response messages from the server (ECU). This supports five different cases (see table below).

Table B1.3 - Physical addressing - periodic transmission

Case	Description
#1	This case specifies **periodically transmitted negative response messages** with response code equal to $78 of the server (ECU) preceded by a request message of the client (tester). **The first response messages is sent within the P2 timing window. For all following negative response messages with response code equal to $78 the P2max* timing becomes active. A detailed specification about the timing behaviour in case of a response code $78 is specified in section 4.1.1 (Timing exceptions).**
#2	This case specifies **periodically transmitted positive response messages (not stopDiagnosticSession and stopCommunication)** of the server (ECU) preceded by a request message (not stopDiagnosticSession and not stopCommunication) of the client (tester). **The response messages are sent within the P2 timing window.**
#3	This case specifies **periodically transmitted negative response messages** with response codes not equal to $78 of the server (ECU) preceded by a request message of the client (tester). **The response messages are sent within the P2 timing window.**

#4	This case specifies **one transmitted stopDiagnosticSession positive response messages** of the server (ECU) preceded by a stopDiagnosticSession request message of the client (tester). The response message is sent within the P2 timing window. After the reception of a the stopDiagnosticSession positive response message the standardDiagnosticModeWithPeriodicTransmission shall be disabled and the default diagnostic session with normal timing and default values shall be active.
#5	This case specifies **one transmitted stopCommunication positive response messages** of the server (ECU) preceded by a stopDiagnosticSession request message of the client (tester). The response message is sent within the P2 timing window. After the reception of a the StopCommunication positive response message the standardDiagnosticModeWithPeriodicTransmission shall be disabled and the default diagnostic session with normal timing and default values shall be active.
#6	This case specifies any transmitted request message of the client (tester) within the P3 timing window. The timing window P3 starts and ends before the timing window P2 during the standardDiagnosticModeWithPeriodicTransmission. As soon as the standardDiagnosticModeWithPeriodicTransmission is activated the communication structure changes and diagnosticModeWithPeriodicTransmission default timing parameter become active. (P3min = 5ms, P3max* = P2min - 5 ms).

2. Functional addressing

2.1 Functional addressing - single positive response message - single server (ECU) addressed

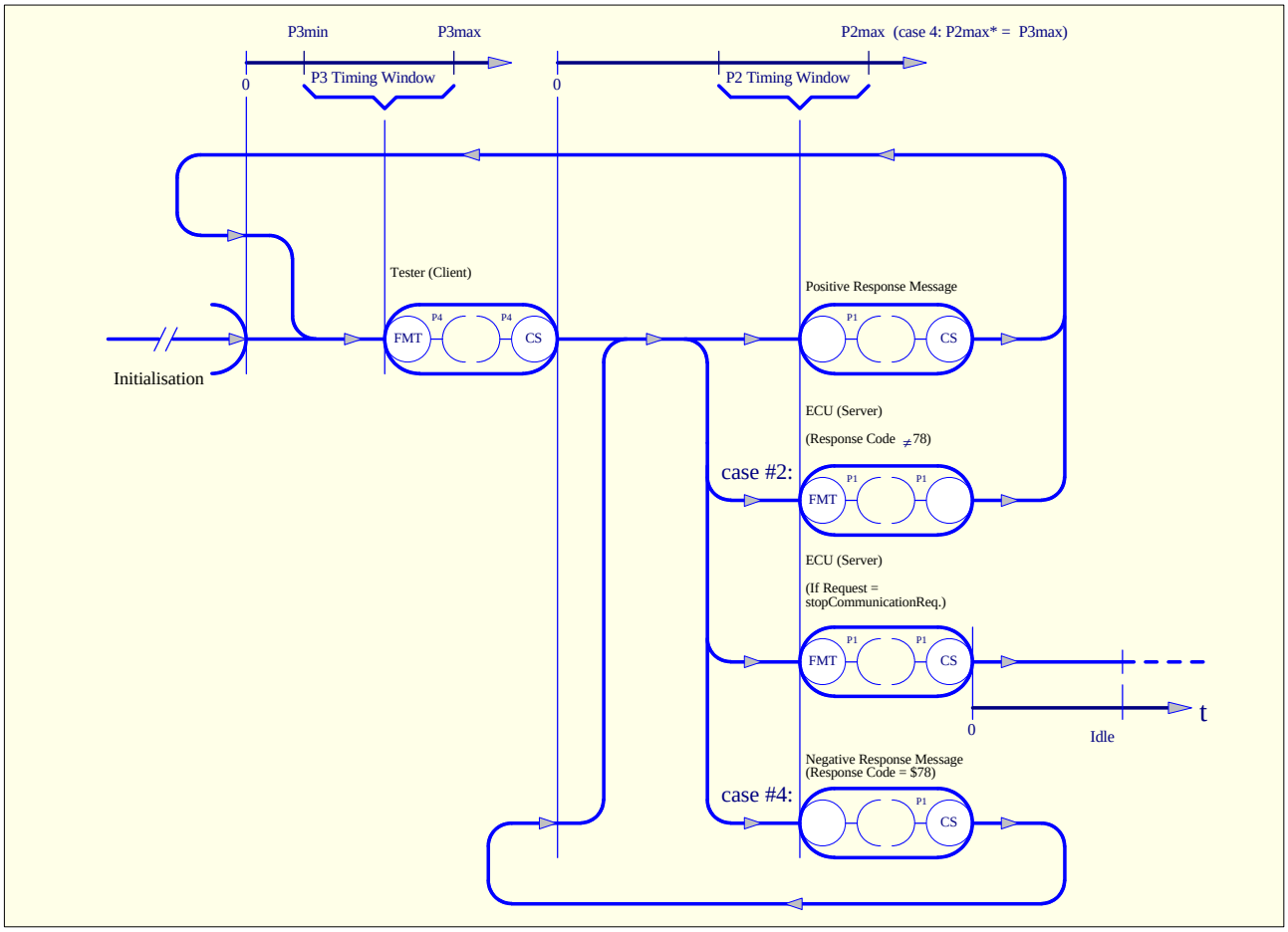


Figure B2.1 - Functional addressing - single positive response message - single server (ECU) addressed

Above figure B2.1 is based on a functional addressing (fast initialisation) followed by a client (tester) request message (target address = single server (ECU)) and a single response message from the server (ECU) addressed. This supports four different cases (see table below).

Table B2.1 - Functional addressing - single positive response message - single server (ECU) addressed

Case	Description
#1	This case specifies a **single positive response message (not stopCommunication)** of the server (ECU) preceded by a request message (not stopCommunication) of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#2	This case specifies a **single negative response message with the negative response code NOT equal to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#3	This case specifies a **single positive stopCommunication response message** of the server (ECU) preceded by a stopCommunication request message of the client (tester). The response message is sent within the P2 timing window. **The P3 time, which has been started after completion of the StopCommunication response message, must first reach the active P3min = T_{Idle} timing value before an "Idle Line" becomes active.**

#4	This case specifies one or multiple **negative response message with the negative response code set to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). **The first response message is sent within the P2 timing window.** Response code '$78' causes the server (ECU) and the client (tester) to modify the P2max timing parameter to change to the P2max*=P3max timing parameter. The client (tester) shall NOT send any request message. This shall provide the server (ECU) more time to prepare for the succeeding response message. The negative response message is followed by another response message as described in case #1, #2. #3 and #4. It depends on the request message of the client (tester) and/or the behaviour of the server (ECU) whether case #1, #2, #3 or #4 becomes active. After completion of case #1, #2 or #3 the P2max*=P3max timing parameter is reset (in both, server (ECU) and client (tester)) to the previous P2max timing parameter after successful completion of the response message. The timing conditions are kept as long as case #4 is repeated. **Note: Case #4 must always be terminated with case #1 or #2 (only if request message = stopCommunication) or #3!**

2.2 Functional addressing - more than one response message - single server (ECU) addressed

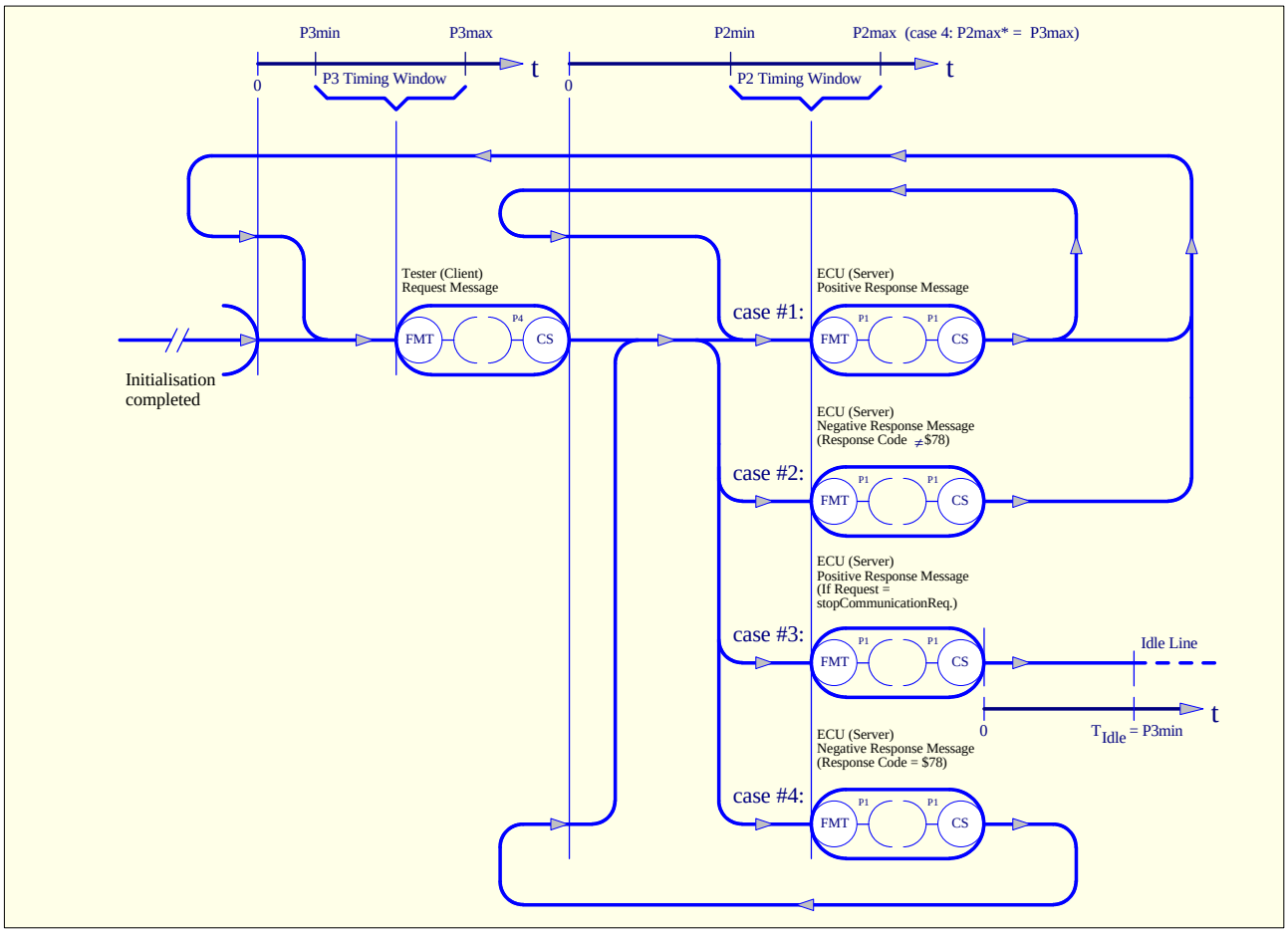


Figure B2.2 - Functional addressing - more than one response message - single server (ECU) addressed

Above figure B2.2 is based on a functional addressing (fast initialisation) followed by a client (tester) request message (target address = single server (ECU)) and more than one response message from the server (ECU) addressed. This supports four different cases (see table below).

Table B2.2 - Functional addressing - more than one response message - single server (ECU) addressed

Case	Description
#1	This case specifies **more than one positive response message (not stopCommunication)** of the server (ECU) preceded by a request message (not stopCommunication) of the client (tester). **The response messages are sent within the P2 timing window. After completion of all positive response messages the last is followed by a client (tester) request message within the P3 timing window.** **Note:** After a positive response message only further positive response messages are allowed (no negative response messages).
#2	This case specifies a **single negative response message with the negative response code NOT equal to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#3	This case specifies a **single positive stopCommunication response message** of the server (ECU) preceded by a stopCommunication request message of the client (tester). The response message is sent within the P2 timing window. **The P3 time, which has been started after completion of the stopCommunication response message, must first reach the active P3min = T_{Idle} timing value before an "Idle Line" becomes active.**

#4	This case specifies one or multiple **negative response message with the negative response code set to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). **The first response message is sent within the P2 timing window.** Response code '$78' causes the server (ECU) and the client (tester) to modify the P2max timing parameter to change to the P2max*=P3max timing parameter. The client (tester) shall NOT send any request message. This shall provide the server (ECU) more time to prepare for the succeeding response message. The negative response message is followed by another response message as described in case #1, #2. #3 and #4. It depends on the request message of the client (tester) and/or the behaviour of the server (ECU) whether case #1, #2, #3 or #4 becomes active. After completion of case #1, #2 or #3 the P2max*=P3max timing parameter is reset (in both, server (ECU) and client (tester)) to the previous P2max timing parameter after successful completion of the response message. The timing conditions are kept as long as case #4 is repeated. **Note: Case #4 must always be terminated with case #1 or #2 or #3 (only if request message = stopCommunication)!**

2.3 Functional addressing - single positive response message - more than one server (ECU)

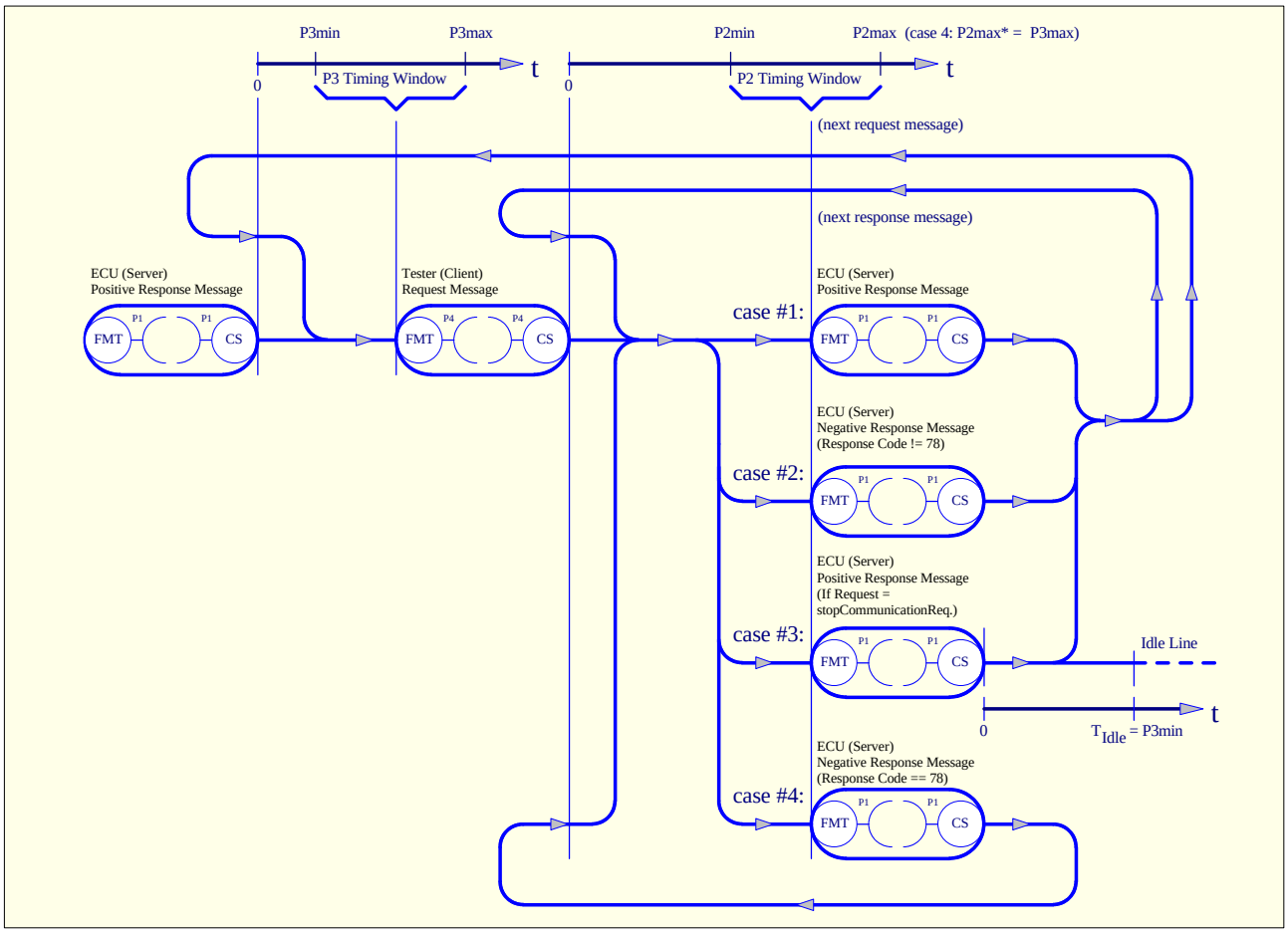


Figure B2.3 - Functional addressing - single positive response message - more than one server (ECU)

Above figure B2.3 is based on a functional addressing (fast initialisation) followed by a client (tester) request message (target address = more than one server (ECU)) and a single response message from each server (ECU) addressed. This supports four different cases (see table below).

Table B2.3 - Functional addressing - single response message - more than one server (ECU)

Case	Description
#1	This case specifies a **single positive response message (not stopCommunication)** of the servers (ECUs) preceded by a request message (not stopCommunication) of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#2	This case specifies a **single negative response message with the negative response code NOT equal to '$78' (requestCorrectlyReceived-ResponsePending)** of the servers (ECUs) preceded by a request message of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#3	This case specifies a **single positive stopCommunication response message** of the servers (ECUs) preceded by a stopCommunication request message of the client (tester). The response message is sent within the P2 timing window. **The P3 time, which has been started after completion of the stopCommunication response message, must first reach the active P3min = T_{Idle} timing value before an "Idle Line" becomes active.**

#4	This case specifies one or multiple **negative response message with the negative response code set to '$78' (requestCorrectlyReceived-ResponsePending)** of the servers (ECUs) preceded by a request message of the client (tester). **The first response message is sent within the P2 timing window.** Response code '$78' causes the server (ECU) and the client (tester) to modify the P2max timing parameter to change to the P2max*=P3max timing parameter. The client (tester) shall NOT send any request message. This shall provide the servers (ECUs) more time to prepare for the succeeding response message. The negative response message is followed by another response message as described in case #1, #2. #3 and #4. It depends on the request message of the client (tester) and/or the behaviour of the servers (ECUs) whether case #1, #2, #3 or #4 becomes active. After completion of case #1, #2 or #3 the P2max*=P3max timing parameter is reset (in both, servers (ECUs) and client (tester)) to the previous P2max timing parameter after successful completion of the response message. The timing conditions are kept as long as case #4 is repeated. **Note: Case #4 must always be terminated with case #1 or #2 or #3 (only if request message = stopCommunication)!**

2.4 Functional addressing - more than one response message - more than one server (ECU)

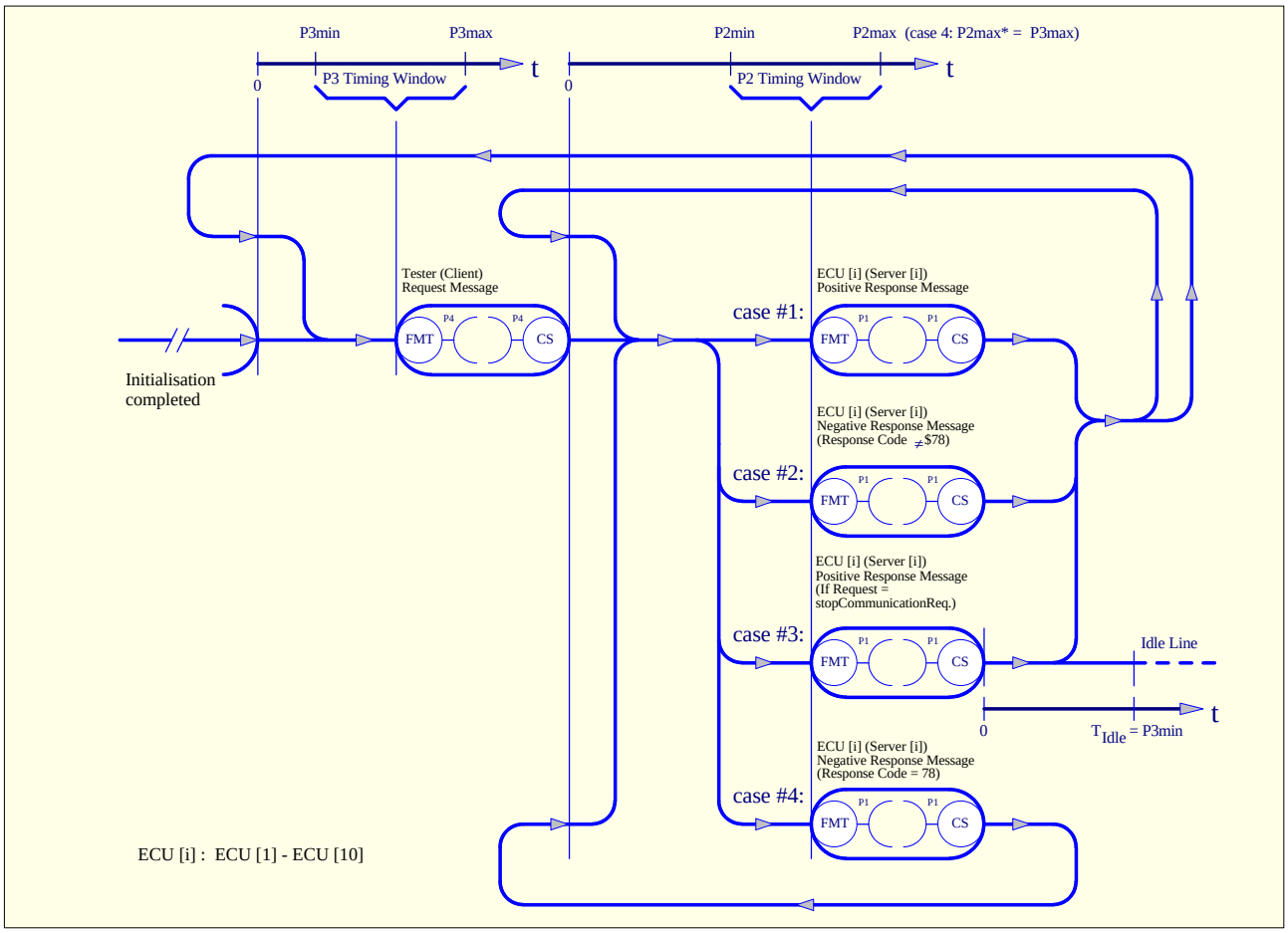


Figure B2.4 - Functional addressing - more than one response message - more than one server (ECU)

Above figure B2.4 is based on a functional addressing (fast initialisation) followed by a client (tester) request message (target address = more than one server (ECU)) and more than 1 response message from the servers (ECUs) addressed. This supports four different cases (see table below).

Table B2.4 - Functional addressing - more than one response message - more than one server (ECU)

Case	Description
#1	This case specifies a **single positive response message (not stopCommunication)** of the server (ECU) preceded by a request message (not stopCommunication) of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window. **Note:** After a positive response message only further positive response messages from the same server (ECU) are allowed (no negative response messages).
#2	This case specifies a **single negative response message with the negative response code NOT equal to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). The response message is sent within the P2 timing window and followed by a client (tester) request message within the P3 timing window.
#3	This case specifies a **single positive stopCommunication response message** of the server (ECU) preceded by a stopCommunication request message of the client (tester). The response message is sent within the P2 timing window. **The P3 time, which has been started after completion of the stopCommunication response message, must first reach the active P3min = T_{Idle} timing value before an "Idle Line" becomes active.**

#4	This case specifies one or multiple **negative response message with the negative response code set to '$78' (requestCorrectlyReceived-ResponsePending)** of the server (ECU) preceded by a request message of the client (tester). **The first response message is sent within the P2 timing window.** Response code '$78' causes the server (ECU) and the client (tester) to modify the P2max timing parameter to change to the P2max*=P3max timing parameter. The client (tester) shall NOT send any request message. This shall provide the server (ECU) more time to prepare for the succeeding response message. The negative response message is followed by another response message as described in case #1, #2. #3 and #4. It depends on the request message of the client (tester) and/or the behaviour of the server (ECU) whether case #1, #2, #3 or #4 becomes active. After completion of case #1, #2 or #3 the P2max*=P3max timing parameter is reset (in both, server (ECU) and client (tester)) to the previous P2max timing parameter after successful completion of the response message. The timing conditions are kept as long as case #4 is repeated. **Note: Case #4 must always be terminated with case #1 or #2 or #3 (only if request message = stopCommunication)!**

Appendix C - Message flow examples

1. Physical initialisation - more than one server (ECU) initialised

This example shows physical initialisation of two ECUs.

STEP#1 startCommunication to ECU1

time	Client (tester) Request Message	Hex
W5	Wake Up Pulse	
	[Fmt = physical addressing Tgt = ECU1] startCommunication.ReqSId	81 11 81

time	Server (ECU1) Positive Response Message	Hex
P2	startCommunication.PosRspSId	C1

STEP#2 e.g. readDataByLocalIdentifier to ECU1

time	Client (tester) Request Message	Hex
P3	readDataByLocalIdentifier[RLI = recordLocalIdentifier]	21 01

time	Server (ECU1) Positive Response Message	Hex	Server (ECU1) Negative Response Message	Hex
P2	readDataByLocalIdentifier.PosRspSId[	61	negativeResponse Service Identifier	7F
	:	01	readDataByLocalIdentifier.ReqSId	21
	:	xx	responseCode	xx

STEP#3 startCommunication to ECU2

time	Client (tester) Request Message	Hex
P3	Wake Up Pulse	
	[Fmt = physical addressing Tgt = ECU2] startCommunication.ReqSId	81 23 81

time	Server (ECU2) Positive Response Message	Hex
P2	startCommunication.PosRspSId	C1

STEP#4 e.g. readDataByLocalIdentifier to ECU1

time	Client (tester) Request Message	Hex
P3	readDataByLocalIdentifier[RLI = recordLocalIdentifier]	21 01

time	Server (ECU1) Positive Response Message	Hex	Server (ECU1) Negative Response Message	Hex
P2	readDataByLocalIdentifier.PosRspSId[	61	negativeResponse Service Identifier	7F
	:	01	readDataByLocalIdentifier.ReqSId	21
	:	xx	responseCode	xx

STEP#5e.g. readDataByLocalIdentifier to ECU2

time	Client (tester) Request Message	Hex
P3	readDataByLocalIdentifier[RLI = recordLocalIdentifier]	21 01

time	Server (ECU2) Positive Response Message	Hex	Server (ECU2) Negative Response Message	Hex
P2	readDataByLocalIdentifier.PosRspSId[	61	negativeResponse Service Identifier	7F
	:	01	readDataByLocalIdentifier.ReqSId	21
	:	xx	responseCode	xx

STEP#6 stopCommunication ECU1

time	Client (tester) Request Message	Hex
P3	stopCommunication.ReqSId	82

time	Server (ECU1) Positive Response Message	Hex	Server (ECU1) Negative Response Message	Hex
P2	stopCommunication.PosRspSId[]	C2	negativeResponse Service Identifier	7F
			stopCommunication.ReqSId[	82
			responseCode]	xx

STEP#7 e.g. readDataByLocalIdentifier to ECU2

time	Client (tester) Request Message	Hex
P3	readDataByLocalIdentifier[RLI = recordLocalIdentifier]	21 01

time	Server (ECU2) Positive Response Message	Hex	Server (ECU2) Negative Response Message	Hex
P2	readDataByLocalIdentifier.PosRspSId[	61	negativeResponse Service Identifier	7F
	:	01	readDataByLocalIdentifier.ReqSId	21
	:	xx	responseCode	xx

STEP#8 stopCommunication ECU2

time	Client (tester) Request Message	Hex
P3	stopCommunication.ReqSId	82

time	Server (ECU2) Positive Response Message	Hex	Server (ECU2) Negative Response Message	Hex
P2	stopCommunication.PosRspSId[]	C2	negativeResponse Service Identifier	7F
			stopCommunication.ReqSId[	82
			responseCode]	xx

2. Periodic Transmission Mode

2.1 Message Flow Example A

This section specifies the conditions to enable the ***Periodic Transmission*** mode ($82) in the client (tester) and the server (ECU).

STEP#1 startDiagnosticSession(DM_PeriodicTransmission)

time	Client (tester) Request Message	Hex
P3	**startDiagnosticSession.ReqSId**[	**10**
	diagnosticMode = PeriodicTransmission DMWPT]	82

time	Server (ECU) Positive Response Message #1	Hex	Server (ECU) Negative Response Message	Hex
P2	startDiagnosticSession.PosRspSId[	**50**	**negativeResponse Service Identifier**	**7F**
	diagnosticMode = DMWPT	82	startDiagnosticSession.ReqSId[	**21**
	PeriodicTransmisson]		responseCode { refer to table 4.4 }]	xx

*** ***PeriodicTransmission mode*** enabled ***
PeriodicTransmission mode default timing values active

time	Server (ECU) Positive Response Message #2	Hex
P2	**startDiagnosticSession.PosRspSId**[	**50**
	diagnosticMode = SDMWPT]	82
●		●
●		●

time	Server (ECU) Positive Response Message #n	Hex
P2	**startDiagnosticSession.PosRspSId**[	**50**
	diagnosticMode = SDMWPT]	82
	goto STEP#2	

STEP#2 e.g. readDataByLocalIdentifier

time	Client (tester) Request Message	Hex
P3*	**readDataByLocalIdentifier**[	**21**
	RLI = recordLocalIdentifier]	01

time	Server (ECU) Positive Response Message #1	Hex	Server (ECU) Negative Response Message	Hex
P2	**readDataByLocalIdentifier.PosRspSId[**	**61**	**negativeResponse Service Identifier**	**7F**
	:	01	**readDataByLocalIdentifier.ReqSId**[	**21**
	:	xx	responseCode { refer to table 4.4 }]	xx
●		●		●
●		●		●

time	Server (ECU) Positive Response Message #n	Hex	Server (ECU) Negative Response Message	Hex
P2	**readDataByLocalIdentifier.PosRspSId[**	**61**	**negativeResponse Service Identifier**	**7F**
	:	01	**readDataByLocalIdentifier.ReqSId**[	**21**
	:	xx	responseCode { refer to table 4.4 }]	xx
	goto STEP#3		**goto STEP#3**	

STEP#3 e.g. stopDiagnosticSession

time	Client (tester) Request Message	Hex
P3*	**stopDiagnosticSession.ReqSId**	**20**

time	Server (ECU) Positive Response Message	Hex	Server (ECU) Negative Response Message	Hex
P2	**stopDiagnosticSession.PosRspSId**[	**60**	**negativeResponse Service Identifier**	**7F**
			stopDiagnosticSession.ReqSId[	**20**
			responseCode { refer to table 4.4 }]	xx

*** ***PeriodicTransmission disabled*** ***
default diagnostic session enabled and
normal timing default values active

*** ***PeriodicTransmission still enabled*** ***

2.2 Message Flow Example B

This section specifies the conditions to enable the ***Periodic Transmission*** mode ($82) in the client (tester) and the server (ECU). It also specifies the use of the service ***accessTimingParameter*** within this mode to modify the ***PeriodicTransmission mode*** default timing.

STEP#1 startDiagnosticSession(DM_PeriodicTransmission)

time	Client (tester) Request Message	Hex
P3	**startDiagnosticSession.ReqSId**[	**10**
	diagnosticMode = PeriodicTransmission SDMWPT]	82

time	Server (ECU) Positive Response Message #1	Hex	Server (ECU) Negative Response Message	Hex
P2	startDiagnosticSession.PosRspSId[	**50**	**negativeResponse Service Identifier**	**7F**
	diagnosticMode = SDMWPT	82	startDiagnosticSession.ReqSId[	**21**
	PeriodicTransmisson]		responseCode { refer to table 4.4 }]	xx

*** ***PeriodicTransmission mode enabled*** ***
PeriodicTransmission mode default timing values active

time	Server (ECU) Positive Response Message #2	Hex
P2	**startDiagnosticSession.PosRspSId**[	**50**
	diagnosticMode = SDMWPT]	82
●		●
●		●

time	Server (ECU) Positive Response Message #n	Hex
P2	**startDiagnosticSession.PosRspSId**[	**50**
	diagnosticMode = SDMWPT]	82
	goto STEP#2	

STEP#2 accessTimingParameters(readLimitsOfPossibleValues)

time	Client (tester) Request Message	Hex
P3*	**accessTimingParameters**[	**83**
	TPI = readLimitsOfPossibleValues]	00

time	Server (ECU) Positive Response Message #2	Hex	Server (ECU) Negative Response Message	Hex
P2	**accessTimingParameters.PosRspSId[**	**C3**	**negativeResponse Service Identifier**	**7F**
	:	xx	**accessTimingParameters.ReqSId**[	**83**
	:	xx	responseCode { refer to table 4.4 }]	xx
●		●		●
●		●		●

time	Server (ECU) Positive Response Message #2	Hex	Server (ECU) Negative Response Message	Hex
P2	**accessTimingParameters.PosRspSId[**	**C3**	**negativeResponse Service Identifier**	**7F**
	TPI = readLimitsOfPossibleValues	00	**accessTimingParameters.ReqSId**[	**83**
	:	xx	responseCode { refer to table 4.4 }]	xx
	goto STEP#3		**return(responseCode)**	

STEP#3 accessTimingParameters(setParameters)

time	Client (tester) Request Message	Hex
P3*	**accessTimingParameters**[	**83**
	TPI = setParameters	03
	:	xx

time	Server (ECU) Positive Response Message #1	Hex	Server (ECU) Negative Response Message	Hex
P2	**accessTimingParameters.PosRspSId[**	**C3**	**negativeResponse Service Identifier**	**7F**
	TPI = setParameters	03	**accessTimingParameters.ReqSId**[	**83**
			responseCode { refer to table 4.4 }]	xx

*** ***modified*** ***timing parameters active*** *** *** ***timing parameters unchanged*** ***

time	Server (ECU) Positive Response Message #2	Hex	Server (ECU) Negative Response Message	Hex
P2	**accessTimingParameters.PosRspSId[**	**C3**	**negativeResponse Service Identifier**	**7F**
	TPI = setParameters]	03	**accessTimingParameters.ReqSId**[	**83**
			responseCode { refer to table 4.4 }]	xx
●		●		●
●		●		●

time	Server (ECU) Positive Response Message #n	Hex	Server (ECU) Negative Response Message	Hex
P2	**accessTimingParameters.PosRspSId[**	**C3**	**negativeResponse Service Identifier**	**7F**
	TPI = setParameters]	03	**accessTimingParameters.ReqSId**[	**83**
			responseCode { refer to table 4.4 }]	xx
	goto STEP#4		**return(responseCode)**	

STEP#4 e.g. readDataByLocalIdentifier

time	Client (tester) Request Message	Hex
P3*	**readDataByLocalIdentifier**[	**21**
	RLI = recordLocalIdentifier]	01

time	Server (ECU) Positive Response Message #1	Hex	Server (ECU) Negative Response Message	Hex
P2	**readDataByLocalIdentifier.PosRspSId[**	**61**	**negativeResponse Service Identifier**	**7F**
	:	xx	**readDataByLocalIdentifierSId**[	**21**
	:	xx	responseCode { refer to table 4.4 }]	xx
●		●		●
●		●		●

time	Server (ECU) Positive Response Message #n	Hex	Server (ECU) Negative Response Message	Hex
P2	**readDataByLocalIdentifier.PosRspSId[**	**61**	**negativeResponse Service Identifier**	**7F**
	:	01	**readDataByLocalIdentifierSId**[	**21**
	:	xx	responseCode { refer to table 4.4 }]	xx
	goto STEP#3		**goto STEP#3**	

STEP#5 e.g. stopDiagnosticSession

time	Client (tester) Request Message	Hex
P3*	**stopDiagnosticSession.ReqSId**	**20**

time	Server (ECU) Positive Response Message	Hex	Server (ECU) Negative Response Message	Hex
P2	**stopDiagnosticSession.PosRspSId**[	**60**	**negativeResponse Service Identifier**	**7F**
			stopDiagnosticSession.ReqSId[	**20**
			responseCode { refer to table 4.4 }]	xx

*** ***PeriodicTransmission disabled*** ***
default diagnostic session enabled and
normal timing default values active

*** ***PeriodicTransmission still enabled*** ***