Appendix 8

CALIBRATION PROTOCOL

CONTENTS

1. INTRODUCTION

This appendix describes how data is exchanged between a vehicle unit and a tester via the K-line which forms part of the calibration interface described in Appendix 6. It also describes control of the input/output signal line on the calibration connector.

Establishing K-line communications is described in Section 4 "Communication Services".

This appendix uses the idea of diagnostic "sessions" to determine the scope of K-line control under different conditions. The default session is the "StandardDiagnosticSession" where all data can be read from a vehicle unit but no data can be written to a vehicle unit.

Selection of the diagnostic session is described in Section 5 "Management Services".

CPR_001 The "ECUProgrammingSession" allows data entry into the vehicle unit. In the case of entry of calibration data (requirements 097 and 098), the vehicle unit must, in addition be in the CALIBRATION mode of operation.

Data transfer via K-line is described in Section 6 "Data Transmission Services". Formats of data transferred are detailed in Section 8 "dataRecords formats".

CPR_002 The "ECUAdjustmentSession" allows the selection of the I/O mode of the calibration I/O signal line via the K-line interface. Control of the calibration I/O signal line is described in section 7 "Control of Test Pulses — Input/Output Control functional unit".

CPR_003 Throughout this document the address of the tester is referred to as 'tt'. Although there may be preferred addresses for testers, the VU shall respond correctly to any tester address. The physical address of the VU is 0xEE.

2. TERMS, DEFINITIONS AND REFERENCES

The protocols, messages and error codes are principally based on the current draft to date of ISO 14229-1 (Road vehicles — Diagnostic systems — Part 1: Diagnostic services, version 6 of 22 February 2001).

Byte encoding and hexadecimal values are used for the service identifiers, the service requests and responses, and the standard parameters.

The term "tester" refers to the equipment used to enter programming/calibration data into the VU.

The terms "client" and "server" refer to the tester and the VU respectively.

The term ECU means "Electronic Control Unit" and refers to the VU.

References:

ISO 14230-2: Road Vehicles — Diagnostic Systems — Keyword Protocol 2000- Part 2: Data Link Layer. First edition: 1999. Vehicles — Diagnostic Systems.

3. OVERVIEW OF SERVICES

3.1. **Services available**

The following table provides an overview of the services that will be available in the recording equipment and are defined in this document.

CPR_004 The table indicates the services that are available in an enabled diagnostic session.

— The first column lists the services that are available,

— the second column includes the section number in this appendix where of service is further defined,

— the third column assigns the assigns the service identifier values for request messages,

— the fourth column specifies the services of the "StandardDiagnosticSession" (SD) which must be implemented in each VU,

— the fifth column specifies the services of the "ECUAdjustmentSession" (ECUAS) which must be implemented to allow control of the I/O signal line in the front panel calibration connector of the VU,

— the sixth column specifies the services of the "ECUProgrammingSession" (ECUPS) which must be implemented to allow for programming of parameters in the VU.

Table 1

Service Identifier value summary table

			Diagnostic Sessions		
Diagnostic Service Name	Section No	SId Req.Value	SD	ECUAS	ECUPS
StartCommunication	4.1	81	■	■	■
StopCommunication	4.2	82	■		
TesterPresent	4.3	3E	■	■	■
StartDiagnosticSession	5.1	10	■	■	■
SecurityAccess	5.2	27	■	■	■
ReadDataByIdentifier	6.1	22	■	■	■
WriteDataByIdentifier	6.2	2E			■
InputOutputControlByIdentifier	7.1	2F		■	

■ This symbol indicates that the service is mandatory in this diagnostic session.
No symbol indicates that this service is not allowed in this diagnostic session.

3.2. **Response codes**

Response codes are defined for each service.

4. COMMUNICATION SERVICES

Some services are necessary to establish and maintain communication. They do not appear on the application layer. The services available are detailed in the following table:

Table 2

Communication services

Service name	Description
StartCommunication	The client requests to start a communication session with a server(s)
StopCommunication	The client requests to stop the current communication session
TesterPresent	The client indicates to the server that it is still present

CPR_005 The StartCommunication Service is used for starting a communication. In order to perform any service, communication must be initialised and the communication parameters need to be appropriate for the desired mode.

4.1. **StartCommunication Service**

CPR_006 Upon receiving a StartCommunication indication primitive, the VU shall check if the requested communication link can be initialised under the present conditions. Valid conditions for the initialisation of a communication link are described in document ISO 14230-2.

CPR_007 Then the VU shall perform all actions necessary to initialise the communication link and send a StartCommunication response primitive with the positive response parameters selected.

CPR_008 If a VU that is already initialised (and has entered any diagnostic session) receives a new StartCommunication request (e.g. due to error recovery in the tester) the request shall be accepted and the VU shall be re-initialised.

CPR_009 If the communication link cannot be initialised for any reason, the VU shall continue operating as it was immediately prior to the attempt to initialise the communication link.

CPR_010 The StartCommunication Request message must be physically addressed.

CPR_011 Initialising the VU for services is performed through a "fast initialisation" method,

— there is a bus-idle time prior to any activity,

— the tester then sends an initialisation pattern,

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

CPR_012 After completion of the initialisation,

— all communication parameters are set to values defined in Table 4 according to the key bytes,

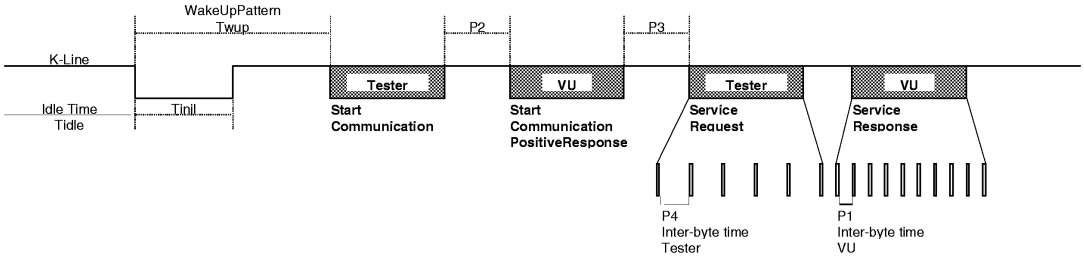
— the VU is waiting for the first request of the tester,

— the VU is in the default diagnostic mode, i.e. StandardDiagnosticSession,

— the calibration I/O signal line is in the default state, i.e. disabled state.

CPR_014 The data rate on the K-line shall be 10 400 Baud.

CPR_016 The fast initialisation is started by the tester transmitting a wake-up pattern (Wup) on the K-line. The pattern begins after the idle time on K-line with a low time of Tinil. The tester transmits the first bit of the StartCommunication Service after a time of Twup following the first falling edge.



CPR_017 The timing values for the fast initialisation and communications in general are detailed in the tables below. There are different possibilities for the idle time:

— first transmission after power on, T_{idle} = 300 ms.

— after completion of a StopCommunication Service, T_{idle} = P3 min.

— after stopping communication by time-out P3 max, T_{idle} = O.

Table 3

Timing values for fast initialisation

Parameter		minimum value	maximum value
Tinil	25 ± 1 ms	24 ms	26 ms
Twup	50 ± 1 ms	49 ms	51 ms

Table 4

Communication timing values

Timing parameter	Parameter description	Lower limit values (ms)	Upper limit values (ms)
		minimum	maximum
P1	Inter byte time for VU response	0	20
P2	Time between tester request and VU response or two VU responses	25	250
P3	Time between end of VU responses and start of new tester request	55	5 000
P4	Inter byte time for tester request	5	20

CPR_018 The message format for fast initialisation is detailed in the following tables:

Table 5

StartCommunication request message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	81	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	StartCommunication Request Service	81	SCR
#5	Checksum	00-FF	CS

Table 6

StartCommunication positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	StartCommunication Positive Response Service Id	C1	SCRPR
#6	Key byte 1	EA	KB1
#7	Key byte 2	8F	KB2
#8	Checksum	00-FF	CS

CPR_019 There is no negative response to the StartCommunication Request message, if there is no positive response message to be transmitted then the VU is not initialised, nothing is transmitted and it remains in its normal operation.

4.2. **StopCommunication service**

4.2.1. ***Message description***

The purpose of this communication layer service is to terminate a communication session.

CPR_020 Upon receiving a StopCommunication indication primitive, the VU shall check if the current conditions allow to terminate this communication. In this case the VU shall perform all actions necessary to terminate this communication.

CPR_021 If it is possible to terminate the communication, the VU shall issue a StopCommunication response primitive with the Positive Response parameters selected, before the communication is terminated.

CPR_022 If the communication cannot be terminated by any reason, the VU shall issue a StopCommunication response primitive with the Negative Response parameter selected.

CPR_023 If time -out of P3max is detected by the VU, the communication shall be terminated without any response primitive being issued.

4.2.2. ***Message format***

CPR_024 The message formats for the StopCommunication primitives are detailed in the following tables:

Table 7

StopCommunication request message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	01	LEN
#5	StopCommunication Request Service Id	82	SPR
#6	Checksum	00-FF	CS

Table 8

StopCommunication positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	01	LEN
#5	StopCommunication Positive Response Service	C2	SPRPR
#6	Checksum	00-FF	CS

Table 9

StopCommunication negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	negative Response Service Id	7F	NR
#6	StopCommunication Request Service Identification	82	SPR
#7	responseCode = generalReject	10	RC_GR
#8	Checksum	00-FF	CS

4.2.3. ***Parameter definition***

This service does not require any parameter definition.

4.3. **TesterPresent service**

4.3.1. ***Message description***

The TesterPresent service is used by the tester to indicate to the server that it is still present, in order to prevent the server from automatically returning to normal operation and possibly stopping the communication. This service, sent periodically, keeps the diagnostic session/communication active by resetting the P3 timer each time a request for this service is received.

4.3.2. ***Message format***

CPR_079 The message formats for the TesterPresent primitives are detailed in the following tables.

Table 10

TesterPresent request message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	02	LEN
#5	TesterPresent Request Service Id	3E	TP
#6	Sub Function = responseRequired = [yes	01	RESPREQ_Y
	no]	02	RESPREQ_NO
#7	Checksum	00-FF	CS

CPR_080 If the responseRequired parameter is set to "yes", then the server shall respond with the following positive response message. If set to "no", then no response is sent by the server.

Table 11

TesterPresent positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	01	LEN
#5	TesterPresent Positive Response Service Id	7E	TPPR
#6	Checksum	00-FF	CS

CPR_081 The service shall support the following negative responses codes:

Table 12

TesterPresent negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	negative Response Service Id	7F	NR
#6	TesterPresent Request Service Identification	3E	TP
#7	responseCode = [SubFunctionNotSupported-InvalidFormat	12	RC_SFNS_IF
	incorrectMessageLength]	13	RC_IML
#8	Checksum	00-FF	CS

5. MANAGEMENT SERVICES

The services available are detailed in the following table:

Table 13

Management services

Service name	Description
StartDiagnosticSession	The client requests to start a diagnostic session with a VU
SecurityAccess	The client requests access to functions restricted to authorised users

5.1. **StartDiagnosticSession service**

5.1.1. ***Message description***

CPR_025 The service StartDiagnosticSession is used to enable different diagnostic sessions in the server. A diagnostic session enables a specific set of services according to Table 17. A session can enable vehicle manufacturer specific services which are not part of this document. Implementation rules shall conform to the following requirements:

— there shall be always exactly one diagnostic session active in the VU,

— the VU shall always start the StandardDiagnosticSession when powered up. If no other diagnostic session is started, then the StandardDiagnosticSession shall be running as long as the VU is powered,

— if a diagnostic session which is already running has been requested by the tester, then the VU shall send a positive response message,

— whenever the tester requests a new diagnostic session, the VU shall first send a StartDiagnosticSession positive response message before the new session becomes active in the VU. If the VU is not able to start the requested new diagnostic session, then it shall respond with a StartDiagnosticSession negative response message, and the current session shall continue.

CPR_026 The diagnostic session shall only be started if communication has been established between the client and the VU.

CPR_027 The timing parameters defined in Table 4 shall be active after a successful StartDiagnosticSession with the diagnosticSession parameter set to “StandardDiagnosticSession” in the request message if another diagnostic session was previously active.

5.1.2. ***Message format***

CPR_028 The message formats for the StartDiagnosticSession primitives are detailed in the following tables:

Table 14

StartDiagnosticSession request message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	02	LEN
#5	StartDiagnosticSession request service Id	10	STDS
#6	diagnosticSession = (one value from Table 17)	xx	DS_…
#7	Checksum	00-FF	CS

Table 15

StartDiagnosticSession positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	02	LEN
#5	StartDiagnosticSession Positive Response Service Id	50	STDSPR
#6	DiagnosticSession = (same value as in byte #6 Table 14)	xx	DS_…
#7	Checksum	00-FF	CS

Table 16

StartDiagnosticSession negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	Negative response service Id	7F	NR
#6	StartDiagnosticSession request service Id	10	STDS
#7	ResponseCode = (subFunctionNotSupported ([a])	12	RC_SFNS
	incorrectMessageLength ([b])	13	RC_IML
	conditionsNotCorrect ([c]))	22	RC_CNC
#8	Checksum	00-FF	CS

([a]) The value inserted in byte #6 of the request message is not supported, i.e. not in Table 17.
([b]) The length of the message is wrong.
([c]) The criteria for the request StartDiagnosticSession are not met.

5.1.3. ***Parameter definition***

CPR_029 The parameter *diagnosticSession (DS_)* is used by the StartDiagnosticSession service to select the specific behaviour of the server(s). The following diagnostic sessions are specified in this document:

Table 17

Definition of diagnosticSession values

Hex	Description	Mnemonic
81	StandardDiagnosticSession This diagnostic session enables all services specified in Table 1 column 4 "SD". These services allow reading of data from a server (VU). This diagnostic Session is active after the initialisation has been successfully completed between client (tester) and server (VU). This diagnostic session may be overwritten by other diagnostic sessions specified in this section.	SD
85	ECUProgrammingSession This diagnostic session enables all services specified in Table 1 column 6 "ECUPS". These services support the memory programming of a server (VU) This diagnostic session may be overwritten by other diagnostic sessions specified in this section.	ECUPS
87	ECUAdjustmentSession This diagnostic session enables all services specified in Table 1 column 5 "ECUAS". These services support the input/output control of a server (VU). This diagnostic session may be overwritten by other diagnostic sessions specified in this section.	ECUAS

5.2. **SecurityAccess service**

Writing of calibration data or access to the calibration input/output line is not possible unless the VU is in CALIBRATION mode. In addition to insertion of a valid workshop card into the VU, it is necessary to enter the appropriate PIN into the VU before access to the CALIBRATION mode is granted.

The SecurityAccess service provides a means to enter the PIN and to indicate to the tester whether or not the VU is in CALIBRATION mode.

It is acceptable that the PIN may be entered through alternative methods.

5.2.1. ***Message description***

The SecurityAccess service consists of a SecurityAccess "requestSeed" message, eventually followed by a SecurityAccess "sendKey" message. The SecurityAccess service must be carried out after the StartDiagnosticSession service.

CPR_033 The tester shall use the SecurityAccess "requestSeed" message to check if the vehicle unit is ready to accept a PIN.

CPR_034 If the vehicle unit is already in CALIBRATION mode, it shall answer the request by sending a "seed" of 0x0000 using the service SecurityAccess Positive Response.

CPR_035 If the vehicle unit is ready to accept a PIN for verification by a workshop card, it shall answer the request by sending a "seed" greater than 0x0000 using the service SecurityAccess positive response.

CPR_036 If the vehicle unit is not ready to accept a PIN from the tester, either because the workshop card inserted is not valid, or because no workshop card has been inserted, or because the vehicle unit expects the PIN from another method, it shall answer the request with a negative response with a response code set to conditionsNotCorrectOrRequestSequenceError.

CPR_037 The tester shall then, eventually, use the SecurityAccess "sendKey" message to forward a PIN to the Vehicle Unit. To allow time for the card authentication process to take place, the VU shall use the negative response code requestCorrectlyReceived-ResponsePending to extend the time to respond. However, the maximum time to respond shall not exceed five minutes. As soon as the requested service has been completed, the VU shall send a positive response message or negative response message with a response code different from this one. The negative response code requestCorrectlyReceived-ResponsePending may be repeated by the VU until the requested service is completed and the final response message is sent.

CPR_038 The vehicle unit shall answer to this request using the service SecurityAccess Positive Response only when in CALIBRATION mode.

CPR_039 In the following cases, the vehicle unit shall answer to this request with a Negative Response with a response code set to:

— subFunctionNot supported: invalid format for the subfunction parameter (accessType),

— conditionsNotCorrectOrRequestSequenceError: vehicle unit not ready to accept a PIN entry,

— invalidKey: PIN not valid and number of PIN checks attempts not exceeded,

— exceededNumberOfAttempts: PIN not valid and number of PIN checks attempts exceeded,

— generalReject: Correct PIN but mutual authentication with workshop card failed.

5.2.2. ***Message format — SecurityAccess — requestSeed***

CPR_040 The message formats for the SecurityAccess "requestSeed" primitives are detailed in the following tables:

Table 18

SecurityAccess request — requestSeed message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	02	LEN
#5	SecurityAccess request service Id	27	SA
#6	accessType — requestSeed	7D	AT_RSD
#7	Checksum	00-FF	CS

Table 19

SecurityAccess — requestSeed positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	04	LEN
#5	SecurityAccess positive response service Id	67	SAPR
#6	accessType — requestSeed	7D	AT_RSD
#7	Seed High	00-FF	SEEDH
#8	Seed Low	00-FF	SEEDL
#9	Checksum	00-FF	CS

Table 20

SecurityAccess negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	negativeResponse Service Id	7F	NR
#6	SecurityAccess request service Id	27	SA
#7	responseCode = (conditionsNotCorrectOrRequestSequenceError	22	RC_CNC
	incorrectMessageLength)	13	RC_IML
#8	Checksum	00-FF	CS

5.2.3. ***Message format — SecurityAccess — sendKey***

CPR_041 The message formats for the SecurityAccess "sendKey" primitives are detailed in the following tables:

Table 21

SecurityAccess request — sendKey message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	m+2	LEN
#5	SecurityAccess Request Service Id	27	SA
#6	accessType — sendKey	7E	AT_SK
#7 to #m+6	Key #1 (High)	xx	KEY
	…	…	
	Key #m (low, m must be a minimum of 4, and a maximum of 8)	xx	
#m+7	Checksum	00-FF	CS

Table 22

SecurityAccess — sendKey positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	02	LEN
#5	SecurityAccess positive response service Id	67	SAPR
#6	accessType — sendKey	7E	AT_SK
#7	Checksum	00-FF	CS

Table 23

SecurityAccess negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	NegativeResponse Service Id	7F	NR
#6	SecurityAccess request service Id	27	SA
#7	ResponseCode = (generalReject	10	RC_GR
	subFunctionNotSupported	12	RC_SFNS
	incorrectMessageLength	13	RC_IML
	conditionsNotCorrectOrRequestSequenceError	22	RC_CNC
	invalidKey	35	RC_IK
	exceededNumberOfAttempts	36	RC_ENA
	requestCorrectlyReceived-ResponsePending)	78	RC_RCR_RP
#8	Checksum	00-FF	CS

6. DATA TRANSMISSION SERVICES

The services available are detailed in the following table:

Table 24

Data transmission services

Service name	Description
ReadDataByIdentifier	The client requests the transmission of the current value of a record with access by recordDataIdentifier
WriteDataByIdentifier	The client requests to write a record accessed by recordDataIdentifier

6.1. **ReadDataByIdentifier service**

6.1.1. ***Message description***

CPR_050 The ReadDataByIdentifier service is used by the client to request data record values from a server. The data are identified by a recordDataIdentifier. It is the VU manufacturer's responsibility that the server conditions are met when performing this service.

6.1.2. ***Message format***

CPR_051 The message formats for the ReadDataByIdentifier primitives are detailed in the following tables:

Table 25

ReadDataByIdentifier request message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	03	LEN
#5	ReadDataByIdentifier Request Service Id	22	RDBI
#6 and #7	recordDataIdentifier = (a valor from Table 28)	xxxx	RDI_…
#8	Checksum	00-FF	CS

Table 26

ReadDataByIdentifier positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	m+3	LEN
#5	ReadDataByIdentifier Positive Response Service Id	62	RDBIPR
#6 and #7	recordDataIdentifier = (the same value as bytes #6 and #7 Table 25)	xxxx	RDI_…
#8 to #m+7	dataRecord() = (data#1 : data#m)	xx : xx	DREC_DATA1 : DREC_DATAm
#m+8	Checksum	00-FF	CS

Table 27

ReadDataByIdentifier negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	NegativeResponse Service Id	7F	NR
#6	ReadDataByIdentifier Request Service Id	22	RDBI
#7	ResponseCode = (requestOutOfRange incorrectMessageLength conditionsNotCorrect)	31 13 22	RC_ROOR RC_IML RC_CNC
#8	Checksum	00-FF	CS

6.1.3. ***Parameter definition***

CPR_052 The parameter recordDataIdentifier (RDI_) in the ReadDataByIdentifier request message identifies a data record.

CPR_053 recordDataIdentifier values defined by this document are shown in the table below.

The recordDataIdentifier table consists of four columns and multiple lines.

— The first column (Hex) includes the "hex value" assigned to the recordDataIdentifier specified in the third column.

— The second column (Data element) specifies the data element of Appendix 1 on which the recordDataIdentifier is based (transcoding is sometimes necessary).

— The third column (Description) specifies the corresponding recordDataIdentifier name.

— The fourth column (Mnemonic) specifies the mnemonic of this recordDataIdentifier.

Table 28

Definition of recordDataIdentifier values

Hex	Data element	recordDataIdentifier Name (see format in Section 8.2)	Mnemonic
F90B	CurrentDateTime	TimeDate	RDI_TD
F912	HighResOdometer	HighResolutionTotalVehicleDistance	RDI_HRTVD
F918	K-ConstantOfRecordingEquipment	Kfactor	RDI_KF
F91C	L-TyreCircumference	LfactorTyreCircumference	RDI_LF
F91D	W-VehicleCharacteristicConstant	WvehicleCharacteristicFactor	RDI_WVCF
F921	TyreSize	TyreSize	RDI_TS
F922	nextCalibrationDate	NextCalibrationDate	RDI_NCD
F92C	SpeedAuthorised	SpeedAuthorised	RDI_SA
F97D	vehicleRegistrationNation	RegisteringMemberState	RDI_RMS
F97E	VehicleRegistrationNumber	VehicleRegistrationNumber	RDI_VRN
F190	VehicleIdentificationNumber	VIN	RDI_VIN

CPR_054 The parameter dataRecord (DREC_) is used by the ReadDataByIdentifier positive response message to provide the data record value identified by the recordDataIdentifier to the client (tester). Data formats are specified in Section 8. Additional user optional dataRecords including VU specific input, internal and output data may be implemented, but are not defined in this document.

6.2. **WriteDataByIdentifier service**

6.2.1. ***Message description***

CPR_056 The WriteDataByIdentifier service is used by the client to write data record values to a server. The data are identified by a recordDataIdentifier. It is the VU manufacturer's responsibility that the server conditions are met when performing this service. To update the parameters listed in Table 28 the VU must be in CALIBRATION mode.

6.2.2. ***Message format***

CPR_057 The message formats for the WriteDataByIdentifier primitives are detailed in the following tables:

Table 29

WriteDataByIdentifier request message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	m+3	LEN
#5	WriteDataByIdentifier request service Id	2E	WDBI
#6 and #7	recordDataIdentifier = (a value from Table 28)	xxxx	RDI_…
#8 to #m+7	dataRecord() = (data#1 : data#m)	xx : xx	DREC_DATA1 : DREC_DATAm
#m+8	Checksum	00-FF	CS

Table 30

WriteDataByIdentifier positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	WriteDataByIdentifier positive response service Id	6E	WDBIPR
#6 and #7	recordDataIdentifier = (the same value as bytes #6 and #7 Table 29)	xxxx	RDI_…
#8	Checksum	00-FF	CS

Table 31

WriteDataByIdentifier negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	NegativeResponse Service Id	7F	NR
#6	WriteDataByIdentifier request service Id	2E	WDBI
#7	ResponseCode = (requestOutOfRange	31	RC_ROOR
	incorrectMessageLength	13	RC_IML
	conditionsNotCorrect)	22	RC_CNC
#8	Checksum	00-FF	CS

6.2.3. ***Parameter definition***

The parameter recordDataIdentifier (RDI_) is defined in Table 28.

The parameter dataRecord (DREC_) is used by the WriteDataByIdentifier request message to provide the data record values identified by the recordDataIdentifier to the server (VU). Data formats are specified in Section 8.

7. CONTROL OF TEST PULSES — INPUT/OUTPUT CONTROL FUNCTIONAL UNIT

The services available are detailed in the following table:

Table 32

Input/Output control functional unit

Service name	Description
InputOutputControlByIdentifier	The client requests the control of an input/output specific to the server

7.1. **Message description**

7.1.1. ***Message description***

There is a connection via the front connector which allows test pulses to be controlled or monitored using a suitable tester.

CPR_058 This calibration I/O signal line can be configured by K-line command using the InputOutputControlByIdentifier service to select the required input or output function for the line. The available states of the line are:

— disabled,

— speedSignalInput, where the calibration I/O signal line is used to input a speed signal (test signal) replacing the motion sensor speed signal,

— realTimeSpeedSignalOutputSensor, where the calibration I/O signal line is used to output the speed signal of the motion sensor,

— RTCOutput, where the calibration I/O signal line is used to output the UTC clock signal.

CPR_059 The vehicle unit must have entered an adjustment session and must be in CALIBRATION mode to configure the state of the line. On exit of the adjustment session or of the CALIBRATION mode the vehicle unit must ensure the calibration I/O signal line is returned to the "disabled" (default) state.

CPR_060 If speed pulses are received at the real time speed signal input line of the VU while the calibration I/O signal line is set to input then the calibration I/O signal line shall be set to output or returned to the disabled state.

CPR_061 The sequence shall be:

— establish communications by StartCommunication Service

— enter an adjustment session by StartDiagnosticSession Service and be in CALIBRATION mode of operation (the order of these two operations is not important).

— change the state of the output by InputOutputControlByIdentifier Service.

7.1.2. ***Message format***

CPR_062 The message formats for the InputOutputControlByIdentifier primitives are detailed in the following tables:

Table 33

InputOutputControlByIdentifier request message

Byte #	Parameter name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	EE	TGT
#3	Source address byte	tt	SRC
#4	Additional length byte	xx	LEN
#5	InputOutputControlByIdentifier request Sid	2F	IOCBI
#6 and #7	InputOutputIdentifier = (CalibrationInputOutput)	F960	IOI_CIO
#8 or #8 to #9	ControlOptionRecord = (inputOutputControlParameter — one value from Table 36 controlState — one value from Table 38 (see note below))	 xx xx	COR_… IOCP_… CS_…
#9 or #10	Checksum	00-FF	CS

Note: The controlState parameter is present only in some cases (see 7.1.3).

Table 34

InputOutputControlByIdentifier positive response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	xx	LEN
#5	inputOutputControlByIdentifier positive response SId	6F	IOCBIPR
#6 and #7	inputOutputIdentifier = (CalibrationInputOutput)	F960	IOI_CIO
#8 or #8 to #9	controlStatusRecord = (		CSR_
	inputOutputControlParameter (same value as byte #8 Table 33)	xx	IOCP_…
	controlState (same value as byte #9 Table 33)) (if applicable)	xx	CS_…
#9 or #10	Checksum	00-FF	CS

Table 35

InputOutputControlByIdentifier negative response message

Byte #	Parameter Name	Hex value	Mnemonic
#1	Format byte — physical addressing	80	FMT
#2	Target address byte	tt	TGT
#3	Source address byte	EE	SRC
#4	Additional length byte	03	LEN
#5	negativeResponse Service Id	7F	NR
#6	inputOutputControlByIdentifier request SId	2F	IOCBI
#7	responseCode = (		
	incorrectMessageLength	13	RC_IML
	conditionsNotCorrect	22	RC_CNC
	requestOutOfRange	31	RC_ROOR
	deviceControlLimitsExceeded)	7A	RC_DCLE
#8	Checksum	00-FF	CS

7.1.3. ***Parameter definition***

CPR_064 The parameter inputOutputControlParameter (IOCP_) is defined in the following table:

Table 36

Definition of inputOutputControlParameter values

Hex	Description	Mnemonic
00	ReturnControlToECU This value shall indicate to the server (VU) that the tester does no longer have control about the calibration I/O signal line.	RCTECU
01	ResetToDefault This value shall indicate to the server (VU) that it is requested to reset the calibration I/O signal line to its default state.	RTD
03	ShortTermAdjustment This value shall indicate to the server (VU) that it is requested to adjust the calibration I/O signal line to the value included in the controlState parameter.	STA

CPR_065 The parameter controlState is present only when the inputOutputControlParameter is set to ShortTermAdjustment and is defined in the following table:

Table 37

Definition of controlState values

Mode	Hex value	Description
Disable	00	I/O line is disabled (default state)
Enable	01	Enable calibration I/O line as speedSignalInput
Enable	02	Enable calibration I/O line as realTimeSpeedSignalOutputSensor
Enable	03	Enable calibration I/O line as RTCOutput

8. DATARECORDS FORMATS

This section details:

— general rules that shall be applied to ranges of parameters transmitted by the vehicle unit to the tester,

— formats that shall be used for data transferred via the Data Transmission Services described in Section 6.

CPR_067 All parameters identified shall be supported by the VU.

CPR_068 Data transmitted by the VU to the tester in response to a request message shall be of the measured type (i.e. current value of the requested parameter as measured or observed by the VU).

8.1. **Transmitted parameter ranges**

CPR_069 Table 38 defines the ranges used to determine the validity of a transmitted parameter.

CPR_070 The values in the range “error indicator” provide a means for the vehicle unit to immediately indicate that valid parametric data is not currently available due to some type of error in the recording equipment.

CPR_071 The values in the range “not available” provide a means for the vehicle unit to transmit a message which contains a parameter that is not available or not supported in that module. The values in the range “not requested” provide a means for a device to transmit a command message and identify those parameters where no response is expected from the receiving device.

CPR_072 If a component failure prevents the transmission of valid data for a parameter, the error indicator as described in Table 38 should be used in place of that parameter's data. However, if the measured or calculated data has yielded a value that is valid yet exceeds the defined parameter range, the error indicator should not be used. The data should be transmitted using the appropriate minimum or maximum parameter value.

Table 38

dataRecords ranges

Range Name	1 byte (Hex value)	2 bytes (Hex value)	4 bytes (Hex value)	ASCII
Valid signal	00 to FA	0000 to FAFF	00000000 to FAFFFFFF	1 to 254
Parameter specific indicator	FB	FB00 to FBFF	FB000000 to FBFFFFFF	none
Reserved range for future indicator bits	FC to FD	FC00 to FDFF	FC000000 to FDFFFFFF	none
Error indicator	FE	FE00 to FEFF	FE000000 to FEFFFFFF	0
Not available or not requested	FF	FF00 to FFFF	FF000000 to FFFFFFFF	FF

CPR_073 For parameters coded in ASCII, the ASCII character "*" is reserved as a delimiter.

8.2. **dataRecords formats**

Table 40 to Table 44 below detail the formats that shall be used via the ReadDataByIdentifier and WriteDataByIdentifier Services.

CPR_074 Table 40 provides the length, resolution and operating range for each parameter identified by its recordDataIdentifier:

Table 39

Format of dataRecords

Parameter Name	Data length (bytes)	Resolution	Operating range
TimeDate	8	See details in Table 40	
HighResolutionTotalVehicleDistance	4	5 m/bit gain, 0 m offset	0 to + 21 055 406 km
Kfactor	2	0,001 pulse/m/bit gain, offset 0	0 to 64,255 pulse/m
LfactorTyreCircumference	2	0,125 10^{-3}/bit gain, 0 offset	0 to 8 031 m
WvehicleCharacteristicFactor	2	0,001 pulse/m/bit gain, 0 offset	0 to 64,255 pulse/m
TyreSize	15	ASCII	ASCII
NextCalibrationDate	3	See details in Table 41	
SpeedAuthorised	2	1/256 km/h/bit gain, 0 offset	0 to 250 996 km/h
RegisteringMemberState	3	ASCII	ASCII
VehicleRegistrationNumber	14	See details in Table 44	
VIN	17	ASCII	ASCII

CPR_075 Table 40 details the formats of the different bytes of the TimeDate parameter:

Table 40

Detailed format of TimeDate (recordDataIdentifier value # F00B)

Byte	Parameter definition	Resolution	Operating range
1	Seconds	0,25 s/bit gain, 0 s offset	0 to 59,75 s
2	Minutes	1 min/bit gain, 0 min offset	0 to 59 min
3	Hours	1 h/bit gain, 0 h offset	0 to 23 h
4	Month	1 month/bit gain, 0 month offset	1 toa 12 month
5	Day	0,25 day/bit gain, 0 day offset (see Note below Table 41)	0,25 to 31,75 day
6	Year	1 year/bit gain, +1985 year offset (see Note below Table 41)	1985 to 2235 year
7	Local Minute Offset	1 min/bit gain, – 125 min offset	– 59 to 59 min
8	Local Hour Offset	1 h/bit gain, – 125 offset	– 23 to + 23 h

CPR_076 Table 41 details the formats of the different bytes of the NextCalibrationDate parameter:

Table 41

Detailed format of NextCalibrationDate (recordDataIdentifier value # F022)

Byte	Parameter definition	Resolution	Operating range
1	Month	1 month/bit gain, 0 month offset	1 to 12 month
2	Day	0,25 day/bit gain, 0 day offset (see Note below)	0,25 to 31,75 day
3	Year	1 Year/bit gain, +1985 year offset (see Note below)	1985 to 2235 year

Note conerning the use of the "Day" parameter:
1. A value of 0 for the date is null. The values 1, 2, 3, and 4 are used to identify the first day of the month; 5, 6, 7, and 8 identify the second day of the month; etc.
2. This parameter does not influence or change the hours parameter above.

Note concerning the use of byte "Year" parameter:
A value of 0 for the year identifies the year 1985; a value of 1 identifies 1986; etc.

CPR_078 Table 42 details the formats of the different bytes of the VehicleRegistrationNumber parameter:

Table 42

Detailed format of VehicleRegistrationNumber (recordDataIdentifier value # F07E)

Byte	Parameter definition	Resolution	Operating range
1	Code Page (as defined in Appendix 1)	ASCII	01 to 0A
2 to 14	Vehicle Registration Number (as defined in Appendix 1)	ASCII	ASCII