MAKING MODERN LIVING POSSIBLE

PVED-CLS Controller For Electrohydraulic Steering

KWP2000 protocol specification

Boot-loader version 3.81, 3.82, 3.85

Revisions and references

Revision history

Date	Change	Revision
16 Jan 2014	Danfoss user documentation template. This document is valid for PVED-CLS application software version 1.72 and later versions.	1.70
12 Aug 2014	Section 1.9 and 1.10 are added. The boot-node ID can be set in the range 00-FEh The KWP2000 protocol will reply to the master node ID	1.71
24 Nov 2014	Information about the boot-loader software version is added to the front page. Section on safety parameterization is added in section 12.	1.72
15 Dec 2014	Checked for compatibility with bootloader 3.82	1.73
27 Jan 2015	Memory address ranges in Section 11 is corrected for checksum calculation purposes.	1.74
07 Apr 2015	Selection between KWP2000 frame padding mode and DLC optimized mode (ISO-15765-2) is added in section 1.11.	1.75
10 June 2016	Examples section elaborated to include normal and failure scenarios of downloading of EEPROM parameters (single, block), Code and Bootloader Errata Section added.	1.76
22 Sept 2016	Security access section updated for sending fixed key and using safe parametrization as security mechanism. Warning added in Request download routine in case of write to EEPROM 0x0h or 0x1 h address location	1.77

Document references

Literature
PVED-CLS communication protocol, L1425546.

Definitions and Abbreviations

CAN	Controller Area Network
EH	1. electro-hydraulic 2. specific type of electro-hydraulic valves
ISO	International Organization for Standardization
ISOBUS	agriculture industry specific communication protocol based on J1939; standardized as ISO 11783
J1939	vehicle bus standard defined by the Society of Automotive Engineers (SAE)
KWP2000	Keyword Protocol 2000, a communication protocol used for on-board vehicle diagnostic systems; standardized as ISO 14230
NOR	Nordborg, Denmark
PGN	Parameter Group Number – identifier defined by the J1939 communication standard
PVED	Proportional Valve Electronic Digital
PVED-CLS	a special type of PVED S5 developed for steering applications
S5	Series-5
SEHS	Safe EH Steering
U8	8-bit long unsigned data type
U16	16-bit long unsigned data type
XID	extended message identifier
PAE	Product Application Engineer
SW	Software
PAE	Product Application Engineer
LSB	Least Significant Byte
MSB	Most Significant Byte
NSB	Next Significant Byte
PCI	Protocol Control Information

Copyright notice

Table of Contents

1 GENERAL INFORMATION 5

1.1 KWP2000 feature overview 5

2 ERRATA INFORMATION 6

3 BOOT-LOADER MODE AND APPLICATION-MODE INTERACTION 6

1.2 Power-up 8
1.3 Boot-loader mode 8
1.4 Application mode 8
1.5 Overview of available KWP2000 services 9

4 NOT FULLY SUPPORTED SERVICES 9

1.6 Security access 9

5 11 BIT AND 29 BIT ADDRESSING MODE 9

6 ADDRESS ASSIGNMENT OF BOOT NODE-IDS 9

1.7 11 bit Identifier format 9
1.8 29 bit Identifier format 10
1.9 Node number range 10
1.10 Single session single Node ID 10
1.11 Frame padding mode / Optimized DLC mode 10

7 BAUD RATE 11

8 PROTOCOL DEFINITION OF THE BOOT-LOADER DATA TELEGRAM 12

9 THE SINGLE FRAME COMMUNICATION 12

10 THE MULTI-MESSAGE COMMUNICATION 13

11 SUPPORTED SERVICES 16

11.1 SI = 10HEX Start_Diagnostic_Session 16
11.2 SI = 1A $_{HEX}$ Read_ECU_Identification 18
11.3 SI = 20 $_{HEX}$ Stop_Diagnostic_Session 19
11.4 SI = 21$_{HEX}$ Request_Service_ID 20
11.5 SI = 27$_{HEX}$ Security_Access 21
11.6 SI = 31$_{HEX}$ Start_Routine_By_Local_Identifier/ Erase_Flash 22
11.7 SI = 31$_{HEX}$ Start_Routine_By_Local_Identifier/ Compare_Checksum 23
11.8 SI = 33$_{HEX}$ Request_Routine_Results_By_Local_Identifier/Erase_flash 24
11.9 SI = 33$_{HEX}$ Request_Routine_Results_By_Local_Id/Compare_Checksum 25
11.10 SI = 34HEX Request_Download/Write 26
11.11 SI = 35$_{HEX}$ Request_Upload/Read 27
11.12 SI = 36$_{HEX}$ Transfer_Data/Read 28
11.13 SI = 36$_{HEX}$ Transfer_Data/Write 29

11.14 SI = 37_{HEX} Request_Transfer_Exit / Write 30

12 SAFE PARAMETERIZATION – DIVERSE PARAMETER DATA UPLOAD 31

13 EXAMPLES 32

13.1 Example 1a-Successfull Setting of Single Parameter in Main with Sector and Signature CRC. 32
13.2 Example 1b-Failure During Setting of Single Parameter in Main with Sector and Signature CRC. 34
13.3 Example 2a-Successfull Setting of parameter data block to the main micro-controller. 36
13.4 Example 2b-Failure During Setting of parameter data block to the main micro-controller. 38
13.5 Example 3a: Successful Downloading Application code to the main micro-controller 39
13.6 Example 3b: Failure during Downloading application code to the main micro-controller 41
13.7 Example 4a: Sucessfull Downloading Boot code to the main micro-controller 43
13.8 Example 4b: Failure during Downloading Boot code to the main micro-controller 45
13.9 Example 5: Multiple nodes access to PVED-CLS 47
13.10 Example 6: Using Official Main Binary (.bios) files for flashing code over KWP. 47

14 ERROR CODES 49

15 FLASHING PROCESS 50

16 PLUS+1 SERVICE TOOL PROTOCOL SUPPORT 50

1 General information

The purpose of this document is to describe the KWP2000 protocol and features which are available in the PVED-CLS. The boot-loader software enables the PVED-CLS to be programmed with program (application software) and data content (parameters) via the CAN bus. The boot-loader software can also be re-programmed if needed.
The main and the safety controller shall be individually programmed via separate boot node IDs.

The following data and software components can read and/or written over the CAN bus:

- Application (flash memory)
- Boot-loader (flash memory)
- Application parameters (EEPROM)

1.1 KWP2000 FEATURE OVERVIEW

- Switching between application mode and boot-loader mode
- Application- and boot-loader software programming. See the flashing procedure in chapter 15.
- Application parameter programming
- Request ECU identification data (hardware and software)
- 11 bit mode or 29 bit mode support
- Use for achieving a CAN-node silent mode

2 Errata information

The latest errata information is always available on the Danfoss homepage via following link:
HTTP://POWERSOLUTIONS.DANFOSS.COM/PRODUCTS/STEERING/PVED-CLS-INTELLIGENT-STEERING-SUB-SYSTEM/
It contains errata information for:

- PVED-CLS boot loader
- PVED-CLS application
- Documentation
- PLUS+1 Service tool
- Other topics related to the steering system

If further information to any errata is required, contact your nearest Danfoss Product Application Engineer

Attention	The system integrator and/or responsible for the target system is advised to periodically observe the errata information as new information will be added as needed.

3 Boot-loader mode and Application-mode interaction

Figure 1 shows the relation between boot-loader mode and application mode.

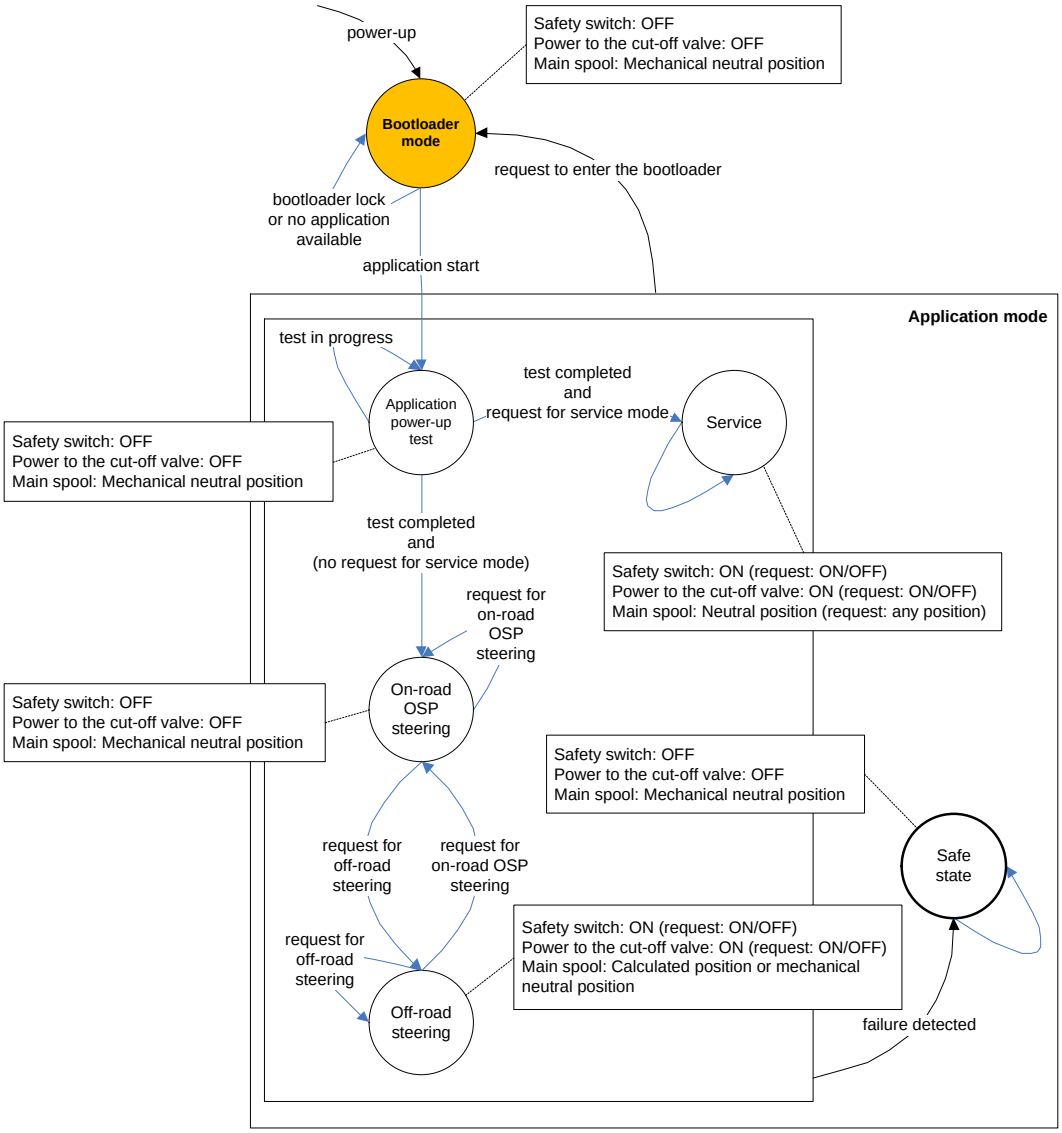


Figure 1 Boot- and Application mode interaction

1.2 POWER-UP

The software enters boot-loader at power-up and checks if valid software is installed. If no valid application software is installed, the PVED-CLS will remain in boot-loader mode.

The boot-loader waits 50ms for a *StartDiagnosticSession* request before a transition to Application mode takes place. The 50ms time window enables a user to force the PVED-CLS in boot-loader mode immediately after power-up and thus avoiding starting the application software.

1.3 BOOT-LOADER MODE

See section 1.5.

1.4 APPLICATION MODE

On transition from boot-loader mode, the application software will begin to execute. The user can request a transition from application mode back to boot-loader mode at any time by means of the *StartDiagnosticSession* service. Refer to PVED-CLS communication protocol, L1425546 for more information on application mode and application communication protocol.

1.5 OVERVIEW OF AVAILABLE KWP2000 SERVICES

Table 1 shows the availability of KWP2000 service in the respective operation mode. The arrows indicate the mode transition that takes place when a specific service is requested.

Service Identifier	Command	Application mode	Boot-loader mode
10_{Hex}	StartDiagnosticSession[1)]	+ →	+
$1A_{Hex}$	ReadECUIdentification	+	+
20_{Hex}	StopDiagnosticSession[1)]	+ ←	+
21_{Hex}	Request Service ID	+	+
27_{Hex}	Security Access	-	+
31_{Hex}	StartRoutineByLocalIdentifier / ERASE_FLASH	-	+
33_{Hex}	RequestRoutineResultsByLocalIdentifier	-	+
34_{Hex}	RequestDownload-Write	-	+
35_{Hex}	RequestUpload-Read	-	+
36_{Hex}	TransferData – Write / Read[2)]	-	+
37_{Hex}	RequestTransferExit – Write / Read[2)]	-	+
31_{Hex}	StartRoutineByLocalIdentifier/COMPARE_CHECKSUM	-	+

Table 1Available KWP2000 services

[1)] These commands also serve for switching between normal and programming modes and can be used for soft-resetting the PVED-CLS.
[2)] Whether Write or Read is to be used, depends on the preceding command (34_{Hex} or 35_{Hex}).

4 Not fully supported services

1.6 SECURITY ACCESS

The PVED-CLS is prepared for security access to support secure sessions between the PVED-CLS and a service tool. The security access service is not fully supported.
A constant key shall be issued prior to accessing parameters
There is no need to request a seed first. Please refer to chapter 11.5 for more information.

5 11 bit and 29 bit addressing mode

The PVED-CLS responds to KWP2000 messages in both 11bit and 29bit mode. It auto-detects the format and continues the operation with the detected message ID format. The two modes should not be mixed during a session.

6 Address assignment of boot node-IDs

In boot-loader mode the PVED-CLS controllers are addressed by their boot node-IDs. The boot-node ID used for KWP2000 sessions and the application node IDs (J1939 source addresses) are programmed via independent parameters. It is possible to set the boot-node ID equal to the application source address.
The address assignment of the nodes is defined in the CAN identifier as follows.

1.7 11 BIT IDENTIFIER FORMAT

CAN ID	101 XXXX XXX Y	Description
Y	Y = 0	Request from programming device
	Y= 1	Response from PVED-CLS
	XXXX XXX	ECU addressing space

Using this assignment method guarantees that all CAN Identifiers are in the $5xx_{Hex}$ range.
The boot node-ID of the ECUs is assigned as follows:

Boot Node ID	Request MSG-ID	Response MSG-ID	ECU
20_{Hex}	540_{Hex}	541_{Hex}	PVED-CLS # main channel
21_{Hex}	542_{Hex}	543_{Hex}	PVED-CLS # safety channel
00_{Hex} – EF_{Hex}	Address range by parameter setting. The value $7F_{Hex}$ must not be used.		

The boot node-ID can be changed via parameter P3 in both main and safety micro-controller.

1.8 29 BIT IDENTIFIER FORMAT

Services with extended (29bit) addressing in Normal Fixed Addressing mode (ISO 15765-2).

Message	CAN identifier
Extended GLOBAL_MSG_ID	18DB32YY $_{Hex}$ YY: Master ID
Extended RX_MSG_ID	18DAXXYY $_{Hex}$ XX: Boot node ID of PVED-CLS YY: Master ID
Extended TX_MSG_ID	18DAYYXX $_{Hex}$ XX: Boot node ID of PVED-CLS YY: Master ID

The message ID has priority 6.

1.9 NODE NUMBER RANGE

The default addresses (node number) of the ECUs are assigned as follows:

Boot Node ID	Request MSG-ID	Response MSG-ID	ECU
20 $_{Hex}$	18DA20F1$_{Hex}$	18DAF120$_{Hex}$	Steering valve controller, main controller
21 $_{Hex}$	18DA21F1$_{Hex}$	18DAF121$_{Hex}$	Steering valve controller, safety controller
00 $_{Hex}$ – EF $_{Hex}$	Address range by parameter setting (P0003)		

The boot node-ID can be changed via parameter P0003 in both main and safety micro-controller. Please note that not all node numbers are valid per ISO 15765 and J1939.
The PVED-CLS replies to any master ID except the boot-node IDs already assigned to the main and safety controller.

1.10 SINGLE SESSION SINGLE NODE ID

The PVED-CLS does not support replying back to more masters in the same session. A session must be finished before another master can access the PVED-CLS. The PVED-CLS will memorize the master's node ID and reply to this address. Requests from other masters are ignored and will not be processed until a new session begins. To start new session stop diagnostics from active master and start diagnostics from the new desired master ECU.
It is not possible to query the PVED-CLS for any ongoing sessions. Refer examples 13.9 for behavior.

1.11 FRAME PADDING MODE / OPTIMIZED DLC MODE

This functionality is available from software version 1.93.
The KWP2000 protocol (ISO-15765-2) allows the use of two different message lengths; a fixed message length (frame padding with DLC=8) and an optimized DLC where the message length is adapted to the number of transmitted data bytes.

P13 Parameter value	PVED-CLS functionality
0 (default)	Optimized DLC mode. The message DLC field of received messages is checked against the received number of data bytes. Transmitted KWP2000 messages have DLC set to the number of transmitted bytes.
255	Frame padding mode. The DLC field for incoming messages is ignored. All transmitted KWP2000 messages have DLC set to 8 and unused bytes set/padded to FF$_{Hex}$.

Frame padding may result in reduced software complexity in interfacing nodes but can lead to a slightly higher bus-load when larger data chunks are transferred. For further information refer to ISO-15765-2.

ATTENTION! Unintended wrong setting of P13 may lead to deadlocks where the configuration tool may not be able to communicate with the PVED-CLS. In this situation use the PLUS+1 service tool to reset P13 to the correct value.

7 Baud rate

250K is the default baud rate used by the system.
The initial Baud Rate can be selected via parameter P5 in both micro-controllers respectively.

ATTENTION!
The boot-loader Software supports 125K, 250K and 1Mbps CAN baud rate. It is recommended always to use 250Kbaud to minimize the complexity of ensuring that other CAN node will tolerate a new baud rate.

8 Protocol definition of the boot-loader data telegram

During the downloading task, larger volumes of data must be transmitted via the CAN bus than is possible with a CAN message, which is limited to 8 data bytes. To bypass this limitation, a data communication layer is set up over CAN layer 2 when programming. Following is a description of two different methods for transporting data on the CAN bus.

The following methods are defined for both cases:
For small data volume less than 8 bytes → the single frame method
For data volume up to 4095 bytes → the multi-message method
Following is an explanation of both of these methods.

9 The single frame communication

This method is relatively simple, since the data volume can be transmitted within a CAN message without a problem and a controlled transmission over several subsequent messages is not necessary.
The procedure is shown below.

Frame Type	PCI														
	Byte #1								Byte #2	Byte #3	Byte #4	Byte #5	Byte #6	Byte #7	Byte #8
PCI name	7	6	5	4	3	2	1	0							
SingleFrame (SF)	0				DL				DATA-0	DATA-1	DATA-2	DATA-3	DATA-4	DATA-5	DATA-6

DL is the frame data length.

10 The multi-message communication

If more than 7 bytes are to be transmitted, the data volume must be distributed over several messages that are sent one after another. At the same time, the receiver has the opportunity to influence the time interval of the telegrams and to inform the sender about the successful or unsuccessful progress of the block transfer. This is done via a so-called "flow control" data telegram.
The following CAN message types have been defined:

First Frame (FF)

Frame Type	**PCI**																	
	Byte #1								**Byte #2**	**Byte #3**	**Byte #4**	**Byte #5**	**Byte #6**	**Byte #7**	**Byte #8**
PCI name	7	6	5	4	3	2	1	0							
First Frame	1				DL-H				DL-L	DATA-0	DATA-1	DATA-2	DATA-3	DATA-4	DATA-5

Consecutive Frame (CF)

Frame Type	**PCI**																	
	Byte #1								**Byte #2**	**Byte #3**	**Byte #4**	**Byte #5**	**Byte #6**	**Byte #7**	**Byte #8**
PCI name	7	6	5	4	3	2	1	0							
Consecutive Frame	2				SN				DATA-0	DATA-1	DATA-2	DATA-3	DATA-4	DATA-5	DATA-6

Flow Control Frame (FC)

Frame Type	**PCI**																	
	Byte #1								**Byte #2**	**Byte #3**	**Byte #4**	**Byte #5**	**Byte #6**	**Byte #7**	**Byte #8**
PCI name	7	6	5	4	3	2	1	0							
Flow Control Frame	3				FS				BS-Max	SP-Min	NA				

DL-H = Frame Data Length High
DL-L = Frame Data Length Low
SN = Sequence Number
FS = Flow Status
BS-Max = Max Block size
SP-Min = Min Seperation Time

The "FirstFrame" (FF) communicates the total number of bytes that belong to the transmission to the sender and already contains the first bytes of the data volume. The rest of the data is transmitted to the receiver via a series of „ConsecutiveFrames" (CF), where a 4-bit message counter serves as a simple data integrity monitor (lost messages). The receiver coordinates the process with the sender via „FlowControlFrames" (FC), with which, among other things, it can inform the sender about its data buffer size (BSMax) and the separation time (in milliseconds) between the individual CFs that it wants. Furthermore, it can also force termination of data transfer with FC frames.

DL (Data Length) is the length of the message in bytes (also for block transfer, separated in high byte and low byte).
SF.DL = 0..7 Bytes (SingleFrame.DataLength)
FF.DL = 0..4095 Bytes (FirstFrame.DataLength)

SN (Sequence Number) is the continuous counter for monitoring data integrity. When receiving the next CF, it must be always be exactly one higher than for the last CF.

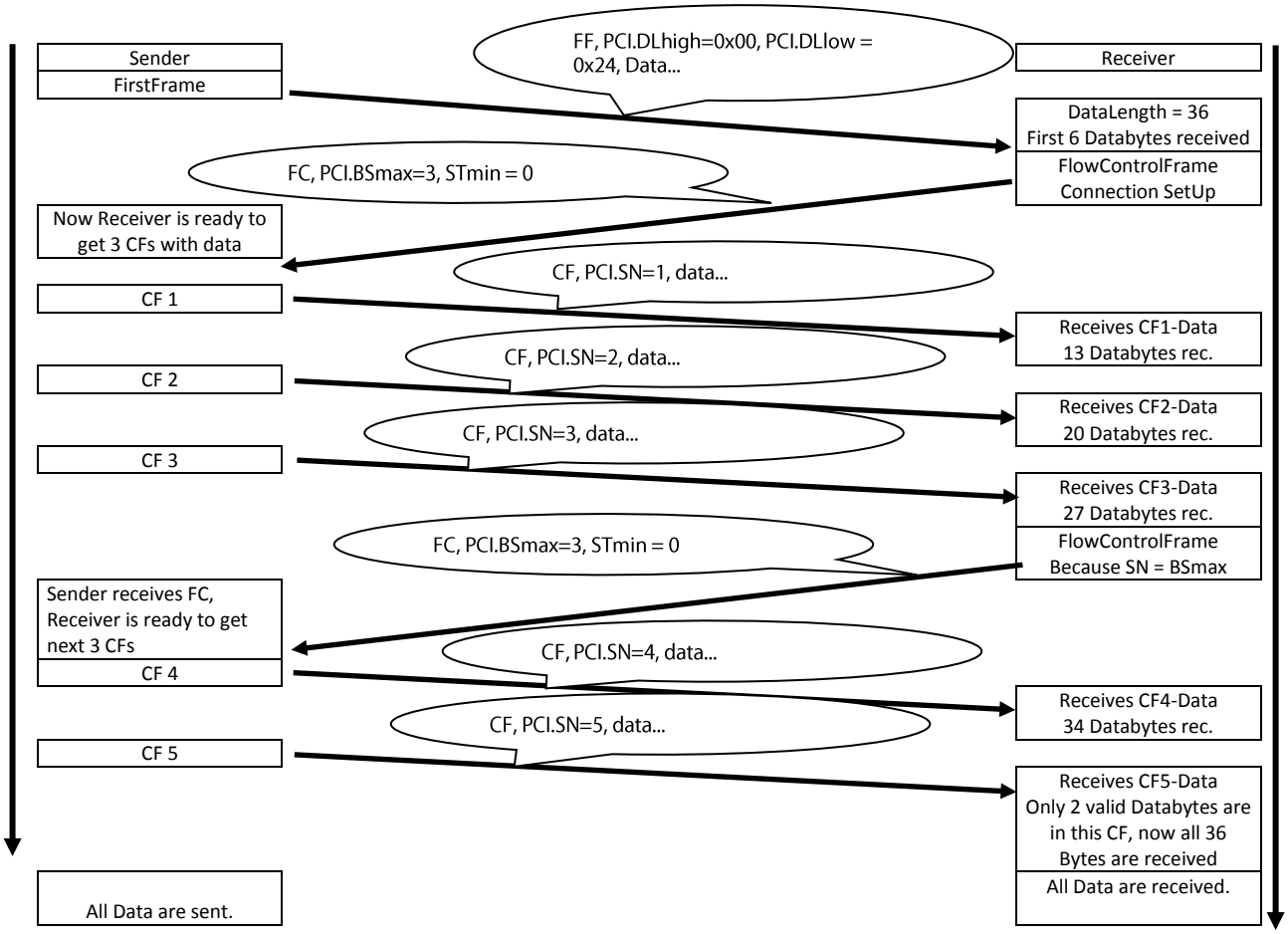
FS (Flow Status) = data for controlling data transfer

PCI.FC.FS nibble				Description of FlowStatus (FS)
3	**2**	**1**	**0**	
0	0	0	0	**ClearToSend (CTS) :** This parameter value is sent to the sender to resume message transmission. Flow control parameters (BSmax, STmin) shall be taken into account by the sender only upon reception of the FlowControl.ClearToSend (FC.CTS) frame that follows the FirstFrame (FF).
0	0	0	1	**WaiT (WT) :** This parameter value is sent by the receiver during the reception of a multiple frame message to inform the sender to pause transmission and wait for an FC.CTS.
0	0	1	0	**RepeatLastMessage(RLM):** This parameter value is sent to the sender to inform that a repeat transmission of the last CAN-Message is necessary. With this FC-Message it is possible to avoid broken data chains because of lost FC- or Acknowledge-Messages [No more bytes needed in CAN-Message]

BSmax = BlockSize. This value indicates how many CAN telegrams have been sent as CF within <u>one</u> protocol frame. It can lie between 0..255 bytes If the data volume to be transferred (encoded with DL) is larger than BSmax*7 (7 data bytes fit in one CF), the data is divided in to (BSmax*7)-sized blocks, which are transmitted one after the other in blocks, delimited by the FC frames.

Special case: 0 means that the receiver is no longer to perform flow control for the received messages.
STmin = Separation Time which is the interval between two CFs in milliseconds.

The multi-message communication procedure is shown in the following schematic:



11 Supported Services

11.1 SI = 10HEX START_DIAGNOSTIC_SESSION

There are a total of three different memory areas available, which can be differentiated by the DCM entry:

Area:	Meaning:	Byte address spaces ECU:
1	Application flash	Address space $7E0000_{Hex}$ to $7FBFFF_{Hex}$
3	Parameter E^2PROM	Address space 0002_{Hex} to 1126_{Hex}
4	Boot-loader flash	Address space $7EC000_{Hex}$ to $7EFFFF_{Hex}$

The following entries are supported by DCM:

DCM_ = 85_{Hex} ECUProgrammingMode
Area 1 (application flash) is enabled for programming.

DCM_ = 84_{Hex} EOLSupplierMode
Area 3 (Parameter E^2PROM) is enabled for programming.

DCM_ = 86_{Hex} ECUDevelopmentMode
Area 4 (boot-loader flash is enabled for programming.

This byte can then be followed by 06_{Hex}. This value, announces a specific network transfer rate, which is then transferred encoded in the three subsequent bytes (MSB, NSB, LSB).
All data fits in one CAN message, so that a SF can be used here.
The complete message is as follows:

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 06_{Hex} (SF, 6 bytes follow at bus speed)
or 02_{Hex} (SF, 2 bytes follow without bus speed)

SI: 10_{Hex} (StartDiagnosticSession)
DCM: 84_{Hex},85_{Hex},86_{Hex} (area)
06_{Hex} (specific Baudrate) [optional]
01_{Hex} (MSB for 100,000 bits/second) [optional]
86_{Hex} (NSB) [optional]
$A0_{Hex}$ (LSB) [optional]

The ECU responds to these requests with a positive or negative response:

ATTENTION!
A positive response is output by the boot-loader software after switching from normal mode to programming mode. This ensures that the end-of-line program knows the current status of the ECU.

The <u>positive response</u> is encoded with a SI = 50_{Hex} so that the message appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 06_{Hex} (SF, 6 bytes follow)
SI: 50_{Hex} (StartDiagnosticSessionPositiveResponse)
DCM: 84_{Hex}, 85_{Hex}, 86_{Hex} (ECU mode)
06_{Hex} (specific Baudrate) [only if request exists]
01_{Hex} (MSB for 100,000 bits/second) [ditto]
86_{Hex} (NSB) [ditto]
$A0_{Hex}$ (LSB) [ditto]

The <u>negative response</u> is encoded with a SI = $7F_{Hex}$, followed by an identifier of the triggered error:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 03_{Hex} (SF, 3 bytes follow)
SI: $7F_{Hex}$ (NegativeResponse)
SDS: 10_{Hex} (on StartDiagnosticSession-SI)
RC: yy_{Hex} (Error condition)
yy = 11_{Hex}: Service not supported
12_{Hex}: Function not supported

Note
When in normal mode, the Start Diagnostic Session message must also be able to be used for switching an ECU into programming mode. To prevent error conditions on the CAN bus, the following must be considered:

1. The CAN bus speed which must be signaled in the Start Diagnostic Session message.

2. If a new bus speed is specified in program mode (boot-loader active), the response message to this command is output at the same speed Start Diagnostic Session message but subsequent messages are transmitted at the requested CAN bus speed.

3. Switching bus speed is only possible if a point-to-point connection exists or all other bus nodes are fully passive on the CAN bus, i.e. they also do not generate acknowledge or error frames.

4. If a new boot-loader software is downloaded in programming mode (boot-loader active), a software reset must be carried out after successful data transmission (checksum OK). After the new boot-loader software has been activated, the ECU returns a positive response with DCM=85.

11.2 SI = 1A $_{HEX}$ READ_ECU_IDENTIFICATION

In response to this message, the ECU that has been addressed sends data about its identification depending on the Identification Option (IO). The following Identification Options (IO) are defined:

IO	Description	Max. length (Bytes)	Memory	Format	Generated by	Function
87 $_{Hex}$	SW-Version (boot loader)	39	FLASH	ASCII	Boot loader	SW identification boot loader
99 $_{Hex}$	Program date (boot loader)	3		BCD-cod.		
8F $_{Hex}$	SW version (application)	39	FLASH	ASCII	Application	SW identification application
90 $_{Hex}$	Program date (application)	3		BCD-cod.		

ATTENTION!

ECU identification must be able to be carried out in normal as well as programming mode.

A positive response is transferred as SF or FF/CF sequence, depending on IO-number and the resulting data length:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x?? (SF/FF)
SI: 5A $_{Hex}$ (PositiveResponse)
IO: 87 $_{Hex}$, 99 $_{Hex}$, 8F $_{Hex}$, 90 $_{Hex}$,… (requested value)
Data ... (all data corresponding to the IO value)

A negative response appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 03 $_{Hex}$ (SF, 3 bytes follow)
SI: 7F $_{Hex}$ (NegativeResponse)
SDS: 1A $_{Hex}$ (on ReadECUIdentification-SI)
RC: yy $_{Hex}$ (Error condition)
yy = 11 $_{Hex}$: Service not supported
12 $_{Hex}$: Function is not supported

Example

18DA20F1$_{Hex}$ = programming device message ID, 18DAF120$_{Hex}$ =PVED-CLS main controller message ID.

MSG ID	DLC	Data (hex)	Time stamp	Comment
18DA20F1X	3	02 1A 87	4237.764580 R	Request ECU identification. 87h = boot-loader SW version.
18DAF120 X	8	10 29 5A 87 42 4F 4F 54	4237.765150 R	PCI.DL = 29h (41 bytes). SI = 5Ah (pos response). IO = 87h Data = 42h, 4Fh, 4F, 54h
18DA20F1 X	3	30 03 00	4237.766240 R	Flow control frame. FS=0h. Max block size = 3 Minimum separation time = 0.
18DAF120 X	8	21 5F 43 4C 53 2D 5F 4D	4237.766820 R	Consecutive frame. Sequence no = 1 Data = 5Fh, 43h, 4Ch, 53h, 2Dh, 5Fh, 4Dh
18DAF120 X	8	22 5F 52 33 37 30 5F 4B	4237.767390 R	Consecutive frame. Sequence no = 2 Data = 5Fh, 52h, 33h, 37h, 30h, 5Fh, 4Bh
18DAF120 X	8	23 57 50 32 30 30 30 2D	4237.767970 R	Consecutive frame. Sequence no = 3 Data = 5Fh, 52h, 33h, 37h, 30h, 5Fh, 4Bh
18DA20F1 X	3	30 03 00	4237.769310 R	Flow control frame. FS=0h. Max block size = 3 Minimum separation time = 0.
18DAF120 X	8	24 5F 30 30 30 30 30 30	4237.769890 R	Consecutive frame. Sequence no = 4. Data = 5Fh, 30h, 30h, 30h, 30h, 30h, 30h
18DAF120 X	8	25 30 30 5F 2D 72 72 72	4237.770460 R	Consecutive frame. Sequence no = 5 Data = 30h, 30h, 5Fh, 2Dh, 72h, 72h, 72h The 41 bytes represents the character string: "BOOT_CLS-_M_R370_KWP2000-_00000000_-rrr"
18DA20F1 X	3	02 1A 99	4237.864480 R	Request ECU identification. 99h = boot-loader SW date
18DAF120 X	6	05 5A 99 07 26 13	4237.864990 R	The data represents the date "072613" (26 July 2013)
18DA20F1 X	3	02 1A 8F	4237.964450 R	Request ECU identification. 8Fh = Application SW version
18DAF120 X	8	10 29 5A 8F 41 50 50 2D	4237.965020 R	
18DA20F1 X	3	30 03 00	4237.966110 R	
18DAF120 X	8	21 5F 43 4C 53 2D 5F 4D	4237.966690 R	
18DAF120 X	8	22 5F 50 31 37 32 5F 53	4237.967260 R	
18DAF120 X	8	23 45 48 53 2D 2D 2D 2D	4237.967780 R	
18DA20F1 X	3	30 03 00	4237.969250 R	
18DAF120 X	8	24 5F 30 30 30 30 30 30	4237.969820 R	The transmitted bytes represents the character string:
18DAF120 X	8	25 30 30 5F 2D 72 72 72	4237.970400 R	"APP-_CLS-_M_P172_SEHS----_00000000_-rrr"
18DA20F1 X	3	02 1A 90	4238.064610 R	Request ECU identification. 8Fh = Application date
18DAF120 X	6	05 5A 90 13 01 14	4238.065120 R	The data represents the date "130114" (13 Jan 2014)

11.3 SI = 20 HEX STOP_DIAGNOSTIC_SESSION

In response to this message, the ECU that has been addressed terminates the programming mode and returns to normal mode (application mode).

The message is encoded as follows:

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 01_{Hex} (SF, 1 byte follows)
SI: 20_{Hex} (ECUReset)

A positive response appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 01_{Hex} (SF, 1 byte follows)
SI: 60_{Hex} (StopDiagnosticSessionPositiveResponse)

A negative response appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 03_{Hex} (SF, 3 further bytes follow)
SI: $7F_{Hex}$ (NegativeResponse)
SDS: 20_{Hex} (on StopDiagnosticSession-SI)
RC: yy_{Hex} (Error condition)
yy = 11_{Hex}: Service not supported
14_{Hex}: Data length error

ATTENTION:

1. If the ECU is in programming mode (boot-loader active), the positive response is output by the firmware after switching over to normal mode. This ensures that the end-of-line program knows the current mode (programming, normal mode).

2. If the ECU is already in normal mode, a positive response is still output.

11.4 SI = 21_{HEX} REQUEST_SERVICE_ID

Request for LocalID. The command will trigger the PVED-CLS main and safety controller to send out their boot node IDs. The service is present in both normal mode and in boot-loader mode.

ATTENTION:

This service is only possible via the fixed CAN-ID $5FE_{HEX}$.

ID: 101 11111110
PCI: 02_{Hex} (SF, 2 further bytes follow)
SI: 21_{Hex} (Request Service ID)
LI: FA_{Hex} (LocalID)

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 03_{Hex} (SF, 3 further bytes follow)
SI: 61_{Hex} (PositiveResponse-SI)
LI: FA_{Hex} (LocalID)
02_{Hex} value

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 03_{Hex} (SF, 3 further bytes follow)
SI: $7F_{Hex}$ (NegativeResponse)
SDS: 21_{Hex} (RequestServiceID-SI)
RC: yy_{Hex} (Error condition)
yy = 11_{Hex}: Service not supported
12_{Hex}: Function is not supported
22_{Hex}: Incorrect conditions

Example

$18DB32F1_{Hex}$ = programming device message ID.

MSG ID	DLC	Data (hex)	Time stamp	Comment
18DB32F1 X	3	02 21 FA	13222.321380 R	Request Service ID from programming device
18DAF120 X	4	03 61 FA 20	13222.321820 R	Positive response from PVED-CLS main controller
18DAF121 X	4	03 61 FA 21	13222.324000 R	Positive response from PVED-CLS safety controller

11.5 SI = 27_{HEX} SECURITY_ACCESS

This service comprises two services – 'Get Seed' and 'Send Key' as described below.

Get Seed

This service is used to get security access seed from the PVED-CLS.

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x02 (SF - 2 Bytes follow)
SI: 0x27 (RequestTransferExit)
LID: 0x01 (Get security Seed)

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x06 (SF, 6 further byte follows)
SI: 0x67 (RequestTransferExit-PositiveResponse-SI)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x27 (on RequestTransferExit -SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported

Send Key

This Service is used to send security Key to ECU.

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x06 (SF, 2 byte follows)
SI: 0x27 (RequestTransferExit)
LID: 0x02 (Send Security Key)
Key: 00, 00, 00, 00

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x02 (SF, 2 further byte follows)
SI: 0x67 (RequestTransferExit-PositiveResponse-SI)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x27 (on RequestTransferExit -SI)
RC: 0xyy (Error condition)
Yy = 0x11: Service not supported

NOTE: The PVED-CLS does not fully support security access. The security access is covered by Safe Parametrization Section 12

11.6 SI = 31_{HEX} START_ROUTINE_BY_LOCAL_IDENTIFIER/ ERASE_FLASH

In response to this message, specific functions for flash programming are called. Which function will be executed is defined by the LocalIdentifier:

LI = 02_{Hex} Start Erase Flash
Erasing of transmitted flash space starts. This command is sent in the following format:

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 10_{Hex} 08 (FF, 8 further bytes follow)
SI: 31_{Hex} (StartRoutineByLocalIdentifier)
LI: 02_{Hex} (Erase Flash)
sa_{Hex} (MSB start address)
sa_{Hex} (NSB start address)
sa_{Hex} (LSB start address)
ea_{Hex} (MSB end address)
ea_{Hex} (NSB end address)
ea_{Hex} (LSB end address)

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 02_{Hex} (SF, 2 further bytes follow)
SI: 71_{Hex} (PositiveResponse-SI)
LI: 02_{Hex} (Erase Flash)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 03_{Hex} (SF, 3 further bytes follow)
SI: $7F_{Hex}$ (NegativeResponse)
SDS: 31_{Hex} (on StartRoutineByLocalIdentifier-SI)
RC: yy_{Hex} (Error condition)
yy = 11_{Hex}: Service not supported
12_{Hex}: Function is not supported
42_{Hex}: Invalid address

Note
Since erasing the flash can take quite some time, it is possible to check the termination of the process using the following service *Request_Routine_Results_by_Local_Identifer*.

11.7 SI = 31HEX START_ROUTINE_BY_LOCAL_IDENTIFIER/ COMPARE_CHECKSUM

In response to this message, specific functions for flash or eeprom programming are called. Which function will be executed is defined by the LocalIdentifier: LI = 01_{Hex} CompareChecksum

The calculation of checksums is started via the specified memory areas. This command is sent in the following format:

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x100A (FF/CF, 10 further bytes follow)
SI: 0x31 (StartRoutineByLocalIdentifier)
LI: 0x01 (CompareChecksum)
0xsa (MSB start address)
0xsa (NSB start address)
0xsa (LSB start address)
0xsa (MSB end address)
0xsa (NSB end address)
0xsa (LSB end address)
0xcs (MSB expected checksum)
0xcs (LSB expected checksum)

A positive response to this command appears as follows:
ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x02 (SF, 2 further bytes follow)
SI: 0x71 (PositiveResponse-SI)
LI: 0x01 (on CompareChecksum)

A negative response to this command appears as follows:
ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x31 (on StartRoutineByLocalIdentifier-SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported
0x12: Function is not supported
0x33: Security access error

Note
The checksum is formed as a sum[modulo16] of the byte-wise addition of the transmitted data values. The result of the checksum comparison shall be queried with the *Request_Routine_Results_by_Local_Identifer* service.

11.8 SI = 33_{HEX} REQUEST_ROUTINE_RESULTS_BY_LOCAL_IDENTIFIER/ERASE_FLASH

Request the results of the routine. This is transmitted in a CAN message as follows:

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x02 (SF, 2 further bytes follow)
SI: 0x33 (RequestRoutineResultsByLocalIdentifier)
LI: 0x02 (Erase Flash)

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x02 (SF, 2 further bytes follow)
SI: 0x73 (RequestRoutineResults_PositiveResponse-SI)
LI: 0x02 (Erase Flash)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x33 (on RequestRoutineResultsByLocalIdentifier -SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported
0x12: Function is not supported
0xFD: Error on Erasing

11.9 SI = 33_{HEX} REQUEST_ROUTINE_RESULTS_BY_LOCAL_ID/COMPARE_CHECKSUM

See section 11 for information on the address ranges for checksum calculations.

Requests the results of the routine. This is transmitted in a CAN message as follows:

ID:	101 TTTTTTT0, where TTTTTTT= ECU address
PCI:	0x02 (SF, 2 further bytes follow)
SI:	0x33 (RequestRoutineResultsByLocalIdentifier)
LI:	0x01 (CompareChecksum)

A positive response to this command appears as follows:

ID:	101 TTTTTTT1, where TTTTTTT= ECU address
PCI:	0x02 (SF, 2 further bytes follow)
SI:	0x73 (RequestRoutineResults_PositiveResponse-SI)
LI:	0x01 (CompareChecksum)

A negative response to this command appears as follows:

ID:	101 TTTTTTT1, where TTTTTTT= ECU address
PCI:	0x03 (SF, 3 further bytes follow)
SI:	0x7F (NegativeResponse)
SDS:	0x33 (on RequestRoutineResultsByLocalIdentifier -SI)
RC:	0xyy (Error condition)
	yy = 0xFA: Checksum Error

11.10 SI = 34HEX REQUEST_DOWNLOAD/WRITE

This message prepares data transfer into flash or EEPROM.

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x1008 (FF, 8 further bytes follow)
SI: 0x34 (RequestDownload)
0xsa (MSB start address)
0xsa (NSB start address)
0xsa (LSB start address)
0x00 = uncompressed data
0xea (MSB memory size)
0xea (NSB memory size)
0xea (LSB memory size)

Memory size = total number of bytes that are to be transferred in this download

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x02 (SF, 2 further bytes follow)
SI: 0x74 (RequestDownload-PositiveResponse-SI)
LI: 0xyy (yy = maximum number of bytes that can be transferred in a block)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x34 (on RequestDownload -SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported
0x12: Function is not supported
0x13: Time-out (300 ms)
0x22: Conditions not correct
0x33: Security access denied
0x42: Invalid address

WARNING: Download/Write to EEPROM address location 0x0h and 0x1h leads to destruction of the application software and PVED-CLS will have to be recovered by forcing bootloader mode directly after boot.

11.11 SI = 35_{HEX} REQUEST_UPLOAD/READ

This message prepares to read out the flash or EEPROM.

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x1008 (FF, 8 further bytes follow)
SI: 0x35 (RequestUpload)
0xsa (MSB start address)
0xsa (NSB start address)
0xsa (LSB start address)
0x00 = uncompressed data
0xea (MSB memory size)
0xea (NSB memory size)
0xea (LSB memory size)
Memory size = total number of bytes that are to be transferred in this download

ATTENTION:

This command is primarily used to read out the data from the EEPROM. For this reason, the transmitted data volume is low. In the current version, this command only supports block transfers of max. 56 bytes. If a larger number is requested, a negative response with the identifier 0x43 (Cannot Download number of data) is returned.

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x02 (SF, 2 further bytes follow)
SI: 0x75 (RequestUpload-PositiveResponse-SI)
LI: 0xyy (yy = total number of bytes that are to be transferred in this download)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x35 (on RequestUpload-SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported
0x12: Function is not supported
0x22: Conditions not correct
0x33: Security access denied
0x42: Invalid address

If this command was executed with a positive response, the actual data transfer can begin.

11.12 SI = 36_{HEX} TRANSFER_DATA/READ

This message transfers data out of the flash or EEPROM.

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x02 (2 Bytes follow)
SI: 0x36 (Transfer data)
0x35 (Read data)

Following this request, the ECU sends a positive response, which contains the desired data, or a negative response, if an error has occurred:

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x10[yy+1] (yy is the number of bytes that are transmitted by the ECU in postive response to the RequestUpload command)
SI: 0x76 (TransferData-PositiveResponse-SI)
yy * Data
....

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x36 (on TransferData -SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported
0x14: Data length error
0x40: Upload finished
0x42: Invalid address
0x43: Cannot download number of data

When the desired number of data has been sent by the ECU, the command is finished. A further terminating routine is not necessary here.

11.13 SI = 36_{HEX} TRANSFER_DATA/WRITE

This message transfers data into flash or EEPROM.

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x10[yy+1] (yy is the number of bytes that are transferred by the ECU in postive response to the RequestDownload command)
SI: 0x36 (TransferData)
yy * Data
....

After each transmitted block, the ECU sends either a positive or negative response.
A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x01 (SF, 1 further byte follows)
SI: 0x76 (TransferData-PositiveResponse-SI)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x36 (on TransferData -SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported
0x13: Time-out (300 ms)
0x22: Conditions not correct
0x33: Security access error
0x40: Download not accepted (finished or aborted)
0x43: Cannot Download number of data
0xFE: ProgrammingError

11.14 SI = 37_{HEX} REQUEST_TRANSFER_EXIT / WRITE

This message terminates data transfer into flash or EEPROM.

ID: 101 TTTTTTT0, where TTTTTTT= ECU address
PCI: 0x01 (SF, 1 byte follows)
SI: 0x37 (RequestTransferExit)

A positive response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x01 (SF, 1 further byte follows)
SI: 0x77 (RequestTransferExit-PositiveResponse-SI)

A negative response to this command appears as follows:

ID: 101 TTTTTTT1, where TTTTTTT= ECU address
PCI: 0x03 (SF, 3 further bytes follow)
SI: 0x7F (NegativeResponse)
SDS: 0x37 (on RequestTransferExit -SI)
RC: 0xyy (Error condition)
yy = 0x11: Service not supported

12 Safe parameterization – diverse parameter data upload

The boot-loader software supports a service for reading out parameter data in a diverse manner. Please note that reading out data diversely does not provide the entire means for achieving safe parameterization. It can however be used to realize a complete diverse software channel for reading out parameter data. This requires implementing additional diverse data handling in a service tool.

- The parameter data is only stored once and protected by CRC.
- Diverse parameter data is only available when upload data and is supported in all read services.
- Diverse parameter data is used to enforce a diverse implementation and display method in a service tool.
- Diverse parameter data means bit-wise inverting the parameter data when forming the reply message.
- Safe parameterization is an optional protocol service for uploading data which shall be used if a safe parameterization method is required.
- The PVED-CLS can be configured without using diverse parameter data upload.
- The boot-loader software will reply "Invalid address" by write attempts to the diverse parameter data memory.

Normal parameter address space encoding is used for the physical memory address range 0 to 00800000_{HEX}.

Diverse parameter address space encoding is used for the physical memory address range above 00800000_{HEX}
The boot-loader recognizes a request for a diverse parameter upload by the most significant bit in byte two of the requested address.

Example: Perform a diverse parameter upload of the parameter data at address 1234_{HEX}
The requesting service tool shall issue an upload request for the data at logical eeprom address: 00**80**1234_{HEX}

13 Examples

13.1 EXAMPLE 1A-SUCCESSFULL SETTING OF SINGLE PARAMETER IN MAIN WITH SECTOR AND SIGNATURE CRC.

This example describes a safe downloading procedure for a single parameter data to EEPROM to Main microcontroller. The procedure is similar for the safety controller except for the boot node id.

MSG ID	DLC	Data(Hex)	Time Stamp	Comments
				Start Diagnostics, Main Microcontroller
18DA20F0 X	3	02 10 84	3666.618320 T	Start diagnostic session on main controller. DCM = 84h (enable eeprom programming)
18DAF020 X	3	02 50 84	3666.618710 R	Positive response from main controller
				Security Access
18DA20F0 X	3	02 27 01	3666.718230 T	Security access - GetSeed
18DAF020 X	7	06 67 01 22 46 00 EC	3666.718800 R	Positive response. On Security access – GetSeed.
18DA20F0 X	7	06 27 02 00 00 00 00	3666.818450 T	Security access – SendKey (fixed 00h 00h 00h 00h)
18DAF020 X	3	02 67 02	3666.818840 R	Positive response on Security access - SendKey
				Request Download, Single Parameter, Start Address (ox0C71h), Size 2bytes.
18DA20F0 X	8	10 08 34 00 0C 71 00 00	3666.918420 T	Request Download - Write. Single parameter Start address 0x0C71h
18DAF020 X	3	30 03 00	3666.918870 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 02	3667.018390 T	Request Download - Write, Memory size 000002h
18DAF020 X	3	02 74 02	3667.018770 R	Positive response on Request Download - Write
				Transfer Write, Single Parameter, Data 0x0400h.
18DA20F0 X	4	03 36 00 04	3667.118290 T	Transfer data - Write, 2 bytes 0x00h (LSB) 04h (MSB) => 0x0400h
18DAF020 X	2	01 76	3667.123480 R	Positive response on Transfer data - Write
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	3667.218260 T	Request Transfer Exit - Write
18DAF020 X	2	01 77	3667.218640 R	Positive response on Request Transfer Exit
				Request Upload, Single Parameter (Start Add: 0x0C71), 2 bytes.
18DA20F0 X	8	10 08 35 00 0C 71 00 00	3667.318420 T	Request Upload - Read, Start address 0x0C71h, uncompressed data
18DAF020 X	3	30 03 00	3667.318870 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 02	3667.418200 T	Request Upload - Read, memory size 000002h
18DAF020 X	3	02 75 02	3667.418640 R	Positive response on Request Upload - Read
				Transfer Read data, Data 0x0400h .
18DA20F0 X	3	02 36 35	3667.518160 T	Transfer data - Read data
18DAF020 X	4	03 76 00 04	3667.518930 R	Positive response, 2 bytes, 0x00h (LSB) 0x40h (MSB) => 0400

				Request Download, Sector CRC, Start Address (0x0C83), Size 2 bytes.
18DA20F0 X	8	10 08 34 00 0C 83 00 00	3667.618390 T	Request Download - Write. CRC Start address 0x0C83h
18DAF020 X	3	30 03 00	3667.618840 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 02	3667.718290 T	Request Download - Write, Memory size 000002h
18DAF020 X	3	02 74 02	3667.718740 R	Positive response on Request Download - Write
				Transfer Write, Sector CRC, Data 0x5961h.
18DA20F0 X	4	03 36 61 59	3667.818320 T	Transfer data - Write, 2 bytes 0x61h (LSB) ,59h (MSB) => 0x5961h
18DAF020 X	2	01 76	3667.823510 R	Positive response on Transfer data - Write
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	3667.918290 T	Request Transfer Exit - Write
18DAF020 X	2	01 77	3667.918680 R	Positive response on Request Transfer Exit
				Request Upload, Sector CRC (Start Add: 0x0C83), 2 bytes.
18DA20F0 X	8	10 08 35 00 0C 83 00 00	3668.018520 T	Request Upload - Read, Start address 0x0C83h, uncompressed data
18DAF020 X	3	30 03 00	3668.018900 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 02	3668.118290 T	Request Upload - Read, memory size 000002h
18DAF020 X	3	02 75 02	3668.118740 R	Positive response on Request Upload - Read
				Transfer Read, Sector CRC, Data 0x5961h.
18DA20F0 X	3	02 36 35	3668.218390 T	Transfer data - Read data
18DAF020 X	4	03 76 61 59	3668.219160 R	Positive response, 2 bytes, 0x61h (LSB) 0x59h (MSB) => 0x5961h
				Request Download, Signature CRC, Start Address (0x0046), Size 2 bytes.
18DA20F0 X	8	10 08 34 00 00 46 00 00	3668.318550 T	Request Download - Write. CRC Start address 0x0046h
18DAF020 X	3	30 03 00	3668.318930 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 02	3668.418390 T	Request Download - Write, Memory size 000002h
18DAF020 X	3	02 74 02	3668.418840 R	Positive response on Request Download - Write
				Transfer Signature CRC, Data 0x2A83h.
18DA20F0 X	4	03 36 83 2A	3668.518420 T	Transfer data - Write, 2 bytes 0x83h (LSB) ,2Ah (MSB) => 0x2A83h
18DAF020 X	2	01 76	3668.523600 R	Positive response on Transfer data - Write
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	3668.618320 T	Request Transfer Exit - Write
18DAF020 X	2	01 77	3668.618770 R	Positive response on Request Transfer Exit
				Request Upload, Signature CRC, Start Add (0x0046h), Size 2 bytes.
18DA20F0 X	8	10 08 35 00 00 46 00 00	3668.718550 T	Request Upload - Read, Start address 0x0046h, uncompressed data
18DAF020 X	3	30 03 00	3668.719000 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 02	3668.818390 T	Request Upload - Read, memory size 000002h
18DAF020 X	3	02 75 02	3668.818840 R	Positive response on Request Upload - Read

				Transfer Read, Signature CRC, Data 0x 2A83h.
18DA20F0 X	3	02 36 35	3668.918420 T	Transfer data - Read data
18DAF020 X	4	03 76 83 2A	3668.919190 R	Positive response, 2 bytes, 0x83h (LSB) 0x2Ah (MSB) => 0x2A83h
				Stop diagnostics, Main Microcontroller
18DA20F0 X	2	01 20	3669.018390 T	Stop Diagnostic session (main controller)
18DAF020 X	2	01 60	3669.018780 R	Positive Response

13.2 EXAMPLE 1B-FAILURE DURING SETTING OF SINGLE PARAMETER IN MAIN WITH SECTOR AND SIGNATURE CRC.

This example describes behavior of system in case of failure during downloading of a single parameter data to EEPROM to Main Microcontroller. The procedure is similar for the safety controller except for the boot node id.

After system reset in case of failure during single parameter download, the system will go to safe state with Main microcontroller sending DTC for invalid EEPROM configuration as shown in below example. The parameter download process shall be re-executed from beginning for successful download of single parameter as described in Example 13.1

MSG ID	DLC	Data (Hex)	Time Stamp	Comments
				Start Diagnostics, Main Microcontroller
18DA20F0 X	3	02 10 84	3666.618320 T	Start diagnostic session on main controller. DCM = 84h (enable eeprom programming)
18DAF020 X	3	02 50 84	3666.618710 R	Positive response from main controller
				Security Access
18DA20F0 X	3	02 27 01	3666.718230 T	Security access - GetSeed
18DAF020 X	7	06 67 01 22 46 00 EC	3666.718800 R	Positive response. On Security access – GetSeed.
18DA20F0 X	7	06 27 02 00 00 00 00	3666.818450 T	Security access – SendKey (fixed 00h 00h 00h 00h)
18DAF020 X	3	02 67 02	3666.818840 R	Positive response on Security access - SendKey
				Request Download, Single Parameter, Start Address (ox0C71h), Size 2bytes.
18DA20F0 X	8	10 08 34 00 0C 71 00 00	3666.918420 T	Request Download - Write. Single parameter Start address 0x0C71h
18DAF020 X	3	30 03 00	3666.918870 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 02	3667.018390 T	Request Download - Write, Memory size 000002h
18DAF020 X	3	02 74 02	3667.018770 R	Positive response on Request Download - Write
				Transfer Write, Single Parameter, Data 0x0400h.
18DA20F0 X	4	03 36 00 04	3667.118290 T	Transfer data - Write, 2 bytes 0x00h (LSB) 04h (MSB) => 0x0400h
18DAF020 X	2	01 76	3667.123480 R	Positive response on Transfer data - Write
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	3667.218260 T	Request Transfer Exit - Write

18DAF020 X	2	01 77	3667.218640 R	Positive response on Request Transfer Exit
				Request Upload, Single Parameter (Start Add: 0x0C71), 2 bytes.
18DA20F0 X	8	10 08 35 00 0C 71 00 00	3667.318420 T	Request Upload - Read, Start address 0x0C71h, uncompressed data
18DAF020 X	3	30 03 00	3667.318870 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 02	3667.418200 T	Request Upload - Read, memory size 000002h
18DAF020 X	3	02 75 02	3667.418640 R	Positive response on Request Upload - Read
				Transfer Read data, Data 0x0400h .
18DA20F0 X	3	02 36 35	3667.518160 T	Transfer data - Read data
18DAF020 X	4	03 76 00 04	3667.518930 R	Positive response, 2 bytes, 0x00h (LSB) 0x40h (MSB) => 0400
				Request Download, Sector CRC, Start Address (0x0C83), Size 2 bytes.
18DA20F0 X	8	10 08 34 00 0C 83 00 00	3667.618390 T	Request Download - Write. CRC Start address 0x0C83h
18DAF020 X	3	30 03 00	3667.618840 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 02	3667.718290 T	Request Download - Write, Memory size 000002h
18DAF020 X	3	02 74 02	3667.718740 R	Positive response on Request Download - Write
				Transfer Write, Sector CRC, Data 0x5961h.
18DA20F0 X	4	03 36 61 59	3667.818320 T	Transfer data - Write, 2 bytes 0x61h (LSB) ,59h (MSB) => 0x5961h
18DAF020 X	2	01 76	3667.823510 R	Positive response on Transfer data - Write
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	3667.918290 T	Request Transfer Exit - Write
18DAF020 X	2	01 77	3667.918680 R	Positive response on Request Transfer Exit
				Request Upload, Sector CRC (Start Add: 0x0C83), 2 bytes.
18DA20F0 X	8	10 08 35 00 0C 83 00 00	3668.018520 T	Request Upload - Read, Start address 0x0C83h, uncompressed data
18DAF020 X	3	30 03 00	3668.018900 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 02	3668.118290 T	Request Upload - Read, memory size 000002h
18DAF020 X	3	02 75 02	3668.118740 R	Positive response on Request Upload - Read
				Transfer Read, Sector CRC, Data 0x5961h.
18DA20F0 X	3	02 36 35	3668.218390 T	Transfer data - Read data
18DAF020 X	4	03 76 61 59	3668.219160 R	Positive response, 2 bytes, 0x61h (LSB) 0x59h (MSB) => 0x5961h

…
Power failure.
Reset the system
...

18FECA5A X	8	10 FF 10 F0 FF 03 FF FF	4.59162 R	DM1 for Ext triggered safe state from Safety Controller.
1CECFF13 X	8	20 0A 00 02 FF CA FE00	4.80506 R	DM1 BAM for Main Controller.
1CEBFF13 X	8	01 10 FF 2B F0 E2 11 2B	4.87508 R	DM1 Incorrect EEPROM for Main Controller.
1CEBFF13 X	8	02 F0 EE 11 FF FF FF FF	4.94516 R	DM1 CRC Failure for Main Controller.

13.3 EXAMPLE 2A-SUCCESSFULL SETTING OF PARAMETER DATA BLOCK TO THE MAIN MICRO-CONTROLLER.

This example describes a safe downloading procedure for a block parameter data to EEPROM to the main microcontroller. The procedure is similar for the safety controller except for the boot node id.

MSG ID	DLC	Data(Hex)	Time Stamp	Comments
				Start Diagnostics, Main Microcontroller
18DA20F0 X	3	02 10 84	9508.927960 T	Start diagnostic session on main controller. DCM = 84h (enable eeprom programming)
18DAF020 X	3	02 50 84	9508.928400 R	Positive response from main controller
				Security Access
18DA20F0 X	3	02 27 01	9509.027990 T	Security access - GetSeed
18DAF020 X	7	06 67 01 22 46 00 28	9509.028560 R	Positive response. On Security access – GetSeed.
18DA20F0 X	7	06 27 02 00 00 00 00	9509.128150 T	Security access – SendKey (fixed 00h 00h 00h 00h)
18DAF020 X	3	02 67 02	9509.128600 R	Positive response on Security access – SendKey
				Request Upload, Sector, Start Address(0x0C71), Size 20 bytes.
18DA20F0 X	8	10 08 35 00 0C 71 00 00	9509.228370 T	Request Upload - Read, Start address 0x0C71h, uncompressed data
18DAF020 X	3	30 03 00	9509.228820 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 14	9509.328280 T	Request Upload - Read, memory size 0000014h
18DAF020 X	3	02 75 14	9509.328720 R	Positive response on Request Upload – Read
				Transfer Read, Sector, Size 20 bytes.
18DA20F0 X	3	02 36 35	9509.427920 T	Transfer data - Read data
18DAF020 X	8	10 15 76 00 05 94 11 C4	9509.430290 R	Positive response, 5 bytes, 0x00h (LSB) 0x05h (MSB) => 0500h
18DA20F0 X	3	30 03 00	9509.527890 T	Flow Control Frame, send next message/block
18DAF020 X	8	21 09 00 00 00 00 00 00	9509.528470 R	Transfer data, 7 bytes (eeprom data)
18DAF020 X	8	22 00 00 00 00 00 00 D8	9509.529040 R	Transfer data, 7 bytes (eeprom data)
18DAF020 X	2	23 00	9509.529430 R	Transfer data, 1 bytes (eeprom data)
				Request Download, Sector, Start Address (0x0C71), Size 20 bytes.
18DA20F0 X	8	10 08 34 00 0C 71 00 00	9509.627990 T	Request Download - Write. parameter Start address 0x0C71h
18DAF020 X	3	30 03 00	9509.628440 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 14	9509.728020 T	Request Download - Write, Memory size 0000014h
18DAF020 X	3	02 74 14	9509.728470 R	Positive response on Request Download – Write
				Transfer Data Write, Sector(Data and CRC), Size 20 bytes.
18DA20F0 X	8	10 15 36 00 05 94 11 C4	9509.828180 T	Transfer data – Write. 20 bytes for transfer. 5 first bytes
18DAF020 X	3	30 03 00	9509.828630 R	Flow Control Frame, send next message/block
18DA20F0 X	3	30 03 00	9509.927890 T	

18DA20F0 X	8	21 09 00 00 00 00 00 00	9510.028180 T	Transfer data, 7 bytes
18DA20F0 X	8	22 00 00 00 00 00 00 D8	9510.128210 T	Transfer data, 7 bytes
18DA20F0 X	2	23 00	9510.227990 T	Transfer data, Last 1 byte
18DAF020 X	2	01 76	9510.243220 R	Positive response on Transfer data
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	9510.328020 T	Request Transfer Exit - Write
18DAF020 X	2	01 77	9510.328400 R	Positive response on Request Transfer Exit
				Request Upload, Sector, Start Address(0x0C71), Size 20 bytes.
18DA20F0 X	8	10 08 35 00 0C 71 00 00	9510.428240 T	Request Upload - Read, Start address 0x0C71h, uncompressed data
18DAF020 X	3	30 03 00	9510.428690 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 14	9510.528150 T	Request Upload - Read, memory size 0000014h
18DAF020 X	3	02 75 14	9510.528600 R	Positive response on Request Upload – Read
				Transfer Read, Sector, Size 20 bytes.
18DA20F0 X	3	02 36 35	9510.628050 T	Transfer data - Read data
18DAF020 X	8	10 15 76 00 05 94 11 C4	9510.630420 R	Positive response, 5 bytes, 0x00h (LSB) 0x05h (MSB) => 0500h
18DA20F0 X	3	30 03 00	9510.728080 T	Flow Control Frame, send next message/block
18DAF020 X	8	21 09 00 00 00 00 00 00	9510.728660 R	Transfer data, 7 bytes (eeprom data)
18DAF020 X	8	22 00 00 00 00 00 00 D8	9510.729240 R	Transfer data, 7 bytes (eeprom data)
18DAF020 X	2	23 00	9510.729620 R	Transfer data, 1 bytes (eeprom data)
				Request Download, Signature CRC, Start Address (0x0046), Size 2 bytes.
18DA20F0 X	8	10 08 34 00 00 46 00 00	9510.828310 T	Request Download - Write. CRC Start address 0x0046h
18DAF020 X	3	30 03 00	9510.828690 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 02	9510.927960 T	Request Download - Write, Memory size 000002h
18DAF020 X	3	02 74 02	9510.928400 R	Positive response on Request Download - Write
				Transfer Signature CRC, Data 0xA884h.
18DA20F0 X	4	03 36 84 A8	9511.027990 T	Transfer data - Write, 2 bytes 0x84h (LSB) ,A8h (MSB) => 0xA884h
18DAF020 X	2	01 76	9511.033170 R	Positive response on Transfer data - Write
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	9511.127960 T	Request Transfer Exit - Write
18DAF020 X	2	01 77	9511.128340 R	Positive response on Request Transfer Exit
				Request Upload, Signature CRC, Start Add (0x0046h), Size 2 bytes.
18DA20F0 X	8	10 08 35 00 00 46 00 00	9511.228310 T	Request Upload - Read, Start address 0x0046h, uncompressed data
18DAF020 X	3	30 03 00	9511.228760 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 02	9511.328150 T	Request Upload - Read, memory size 000002h
18DAF020 X	3	02 75 02	9511.328600 R	Positive response on Request Upload - Read

				Transfer Read, Signature CRC, Data 0x A884h.
18DA20F0 X	3	02 36 35	9511.428180 T	Transfer data - Read data
18DAF020 X	4	03 76 84 A8	9511.428880 R	Positive response, 2 bytes, 0xA8h (LSB) 0xA8h (MSB) => 0xA884h
				Stop diagnostics
18DA20F0 X	2	01 20	9511.528020 T	Stop Diagnostic session (main controller)
18DAF020 X	2	01 60	9511.528460 R	Positive Response

13.4 EXAMPLE 2B-FAILURE DURING SETTING OF PARAMETER DATA BLOCK TO THE MAIN MICRO-CONTROLLER.

This example describes behavior of system in case of failure during downloading of a block parameter data to EEPROM for main microcontroller. The procedure is similar for the safety controller except for the boot node id.

After system reset in case of failure during block parameter download, the system will go to safe state with Main microcontroller sending DTC for invalid EEPROM configuration as shown in below example. The parameter download process shall be re-executed from beginning for successful download of block parameter as described in Example 13.3

MSG ID	DLC	Data(Hex)	Time Stamp	Comments
				Start Diagnostics, Main Microcontroller
18DA20F0 X	3	02 10 84	9508.927960 T	Start diagnostic session on main controller. DCM = 84h (enable eeprom programming)
18DAF020 X	3	02 50 84	9508.928400 R	Positive response from main controller
				Security Access
18DA20F0 X	3	02 27 01	9509.027990 T	Security access - GetSeed
18DAF020 X	7	06 67 01 22 46 00 28	9509.028560 R	Positive response. On Security access – GetSeed.
18DA20F0 X	7	06 27 02 00 00 00 00	9509.128150 T	Security access – SendKey (fixed 00h 00h 00h 00h)
18DAF020 X	3	02 67 02	9509.128600 R	Positive response on Security access – SendKey
				Request Upload, Sector, Start Address(0x0C71), Size 20 bytes.
18DA20F0 X	8	10 08 35 00 0C 71 00 00	9509.228370 T	Request Upload - Read, Start address 0x0C71h, uncompressed data
18DAF020 X	3	30 03 00	9509.228820 R	Flow Control Frame (send next message)
18DA20F0 X	3	21 00 14	9509.328280 T	Request Upload - Read, memory size 0000014h
18DAF020 X	3	02 75 14	9509.328720 R	Positive response on Request Upload – Read
				Transfer Read, Sector, Size 20 bytes.
18DA20F0 X	3	02 36 35	9509.427920 T	Transfer data - Read data

18DAF020 X	8	10 15 76 00 05 94 11 C4	9509.430290 R	Positive response, 5 bytes, 0x00h (LSB) 0x05h (MSB) => 0500h
18DA20F0 X	3	30 03 00	9509.527890 T	Flow Control Frame, send next message/block
18DAF020 X	8	21 09 00 00 00 00 00 00	9509.528470 R	Transfer data, 7 bytes (eeprom data)
18DAF020 X	8	22 00 00 00 00 00 00 D8	9509.529040 R	Transfer data, 7 bytes (eeprom data)
18DAF020 X	2	23 00	9509.529430 R	Transfer data, 1 bytes (eeprom data)
				Request Download, Sector, Start Address (0x0C71), Size 20 bytes.
18DA20F0 X	8	10 08 34 00 0C 71 00 00	9509.627990 T	Request Download - Write. parameter Start address 0x0C71h
18DAF020 X	3	30 03 00	9509.628440 R	Flow Control Frame, send next message/block
18DA20F0 X	3	21 00 14	9509.728020 T	Request Download - Write, Memory size 0000014h
18DAF020 X	3	02 74 14	9509.728470 R	Positive response on Request Download – Write
				Transfer Data Write, Sector(Data and CRC), Size 20 bytes.
18DA20F0 X	8	10 15 36 00 05 94 11 C4	9509.828180 T	Transfer data – Write. 20 bytes for transfer. 5 first bytes
18DAF020 X	3	30 03 00	9509.828630 R	Flow Control Frame, send next message/block
18DA20F0 X	3	30 03 00	9509.927890 T	
18DA20F0 X	8	21 09 00 00 00 00 00 00	9510.028180 T	Transfer data, 7 bytes
18DA20F0 X	8	22 00 00 00 00 00 00 D8	9510.128210 T	Transfer data, 7 bytes
18DA20F0 X	2	23 00	9510.227990 T	Transfer data, Last 1 byte
18DAF020 X	2	01 76	9510.243220 R	Positive response on Transfer data
				Request Transfer Exit - Write
18DA20F0 X	2	01 37	9510.328020 T	Request Transfer Exit - Write
18DAF020 X	2	01 77	9510.328400 R	Positive response on Request Transfer Exit

…
Power failure.
Reset the system
...

18FECA5A X	8	10 FF 10 F0 FF 03 FF FF	4.59162 R	DM1 for Ext triggered safe state from Safety Controller.
1CECFF13 X	8	20 0A 00 02 FF CA FE00	4.80506 R	DM1 BAM for Main Controller.
1CEBFF13 X	8	01 10 FF 2B F0 E2 11 2B	4.87508 R	DM1 Incorrect EEPROM for Main Controller.
1CEBFF13 X	8	02 F0 EE 11 FF FF FF FF	4.94516 R	DM1 CRC Failure for Main Controller.

13.5 EXAMPLE 3A: SUCCESSFUL DOWNLOADING APPLICATION CODE TO THE MAIN MICRO-CONTROLLER

This example describes the procedure for downloading application code to main controller flash memory.
The procedure is similar for the safety controller except for the boot node id.

Setting one of the controllers in boot-loader mode will automatically set the other controller is boot-loader mode as well.

MSG ID	DLC	Data (hex)	Time stamp	Comment
18DA20F1 X	7	06 10 85 06 03 D0 90	18692.776860 R	Start diagnostic session. DCM = 85h (enable flash programming).

				Network transfer rate = 250K.
18DAF120 X	7	06 50 85 06 03 D0 90	18692.776290 R	Positive response.
18DA20F1 X	3	02 27 01	18692.775200 R	Security access - GetSeed
18DAF120 X	7	06 67 01 06 03 D0 9A	18692.771810 R	Positive response. On Security access – GetSeed. Seed = 06h 03h D0h 9Ah
18DA20F1 X	7	06 27 02 00 00 00 00	18692.771040 R	Security access – SendKey (fixed 00h 00h 00h 00h)
18DAF120 X	3	02 67 02	18692.770590 R	Positive response on Security access - SendKey
				Erase flash
18DA20F1 X	8	10 08 31 02 7E 00 00 7F	18692.768930 R	Start Routine By Local Identifier - Erase flash, start address 7E0000h
18DAF120 X	3	30 03 00	18692.768480 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 BF FF	18692.767070 R	Start Routine By Local Identifier - Erase flash, end address 7FBFFFh
18DAF120 X	3	02 71 02	18692.762210 R	Positive response on erase flash
18DA20F1 X	3	02 33 02	18692.761630 R	Request routine results by local identifier on erase flash
18DAF120 X	3	02 73 02	18687.135840 R	Positive response
				Request Download Write
18DA20F1 X	8	10 08 34 7E 00 00 00 01	18687.134940 R	Request Download - Write. Start address 7E00000h
18DAF120 X	3	30 03 00	18687.134560 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 C0 00	18687.133150 R	Request download – Write. Size in bytes 01C000h
18DAF120 X	3	02 74 38	18687.132700 R	Positive response on request download
				Transfer data – Write (56 byte block)
18DA20F1 X	8	10 39 36 56 1F 76 22 B9	18687.130910 R	Transfer data – Write. 56 bytes for transfer. 6 first bytes
18DAF120 X	3	30 03 00	18687.130460 R	Flow Control Frame, send next message/block
18DA20F1 X	8	21 C0 28 29 00 68 76 1A	18687.128540 R	Transfer data, 7 bytes
18DA20F1 X	8	22 00 7F 23 04 22 57 00	18687.127900 R	Transfer data, 7 bytes
18DA20F1 X	8	23 3F 22 57 00 3F 22 57	18687.127390 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	18687.126940 R	Flow Control Frame, send next message/block
18DA20F1 X	8	24 00 3F 22 57 00 3F 22	18687.125660 R	Transfer data. 7 bytes
18DA20F1 X	8	25 57 00 3F 22 57 00 3F	18687.125090 R	Transfer data. 7 bytes
18DA20F1 X	8	26 22 57 00 3F 22 57 00	18687.124510 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	18687.124060 R	Flow Control Frame, send next message/block
18DA20F1 X	8	27 3F 22 57 00 3F 22 57	18687.122460 R	Transfer data. 7 bytes
18DA20F1 X	3	28 00 3F	18687.122080 R	Transfer data. Last two bytes
18DAF120 X	2	01 76	18687.120800 R	Positive response on Transfer data

…

Data removed to reduce example size.

...

18DA20F1 X	8	27 FF FF FF FF FF FF FF	18664.349020 R	Last 56 block transfer
18DA20F1 X	3	28 FF FF	18664.348640 R	

18DAF120 X	2	01 76	18664.347870 R	
18DA20F1 X	8	10 39 36 FF FF FF FF FF	18664.347100 R	
18DAF120 X	3	30 03 00	18664.346660 R	
18DA20F1 X	8	21 FF FF FF FF FF FF FF	18664.344990 R	
18DA20F1 X	8	22 FF FF FF FF FF FF FF	18664.344420 R	
18DA20F1 X	8	23 FF FF FF 00 41 00 50	18664.343780 R	
18DAF120 X	3	30 03 00	18664.343390 R	
18DA20F1 X	8	24 00 50 00 2D 00 5F 00	18664.341920 R	
18DA20F1 X	8	25 43 00 4C 00 53 00 2D	18664.341340 R	
18DA20F1 X	8	26 00 5F 00 4D 00 5F 00	18664.340770 R	
18DAF120 X	3	30 03 00	18664.340320 R	
18DA20F1 X	8	27 50 00 31 00 37 00 32	18664.338780 R	
18DA20F1 X	3	28 00 5F	18664.338400 R	
18DAF120 X	2	01 76	18664.337060 R	End of last 56 bytes block
18DA20F1 X	2	01 37	18664.324580 R	Request transfer exit – Write
18DAF120 X	2	01 77	18664.324190 R	Positive response on Request transfer exit
18DA20F1 X	8	10 0A 31 01 7E 00 00 7F	18664.323300 R	Start routine by local identifier. Compare checksum. Start address 7E0000h
18DAF120 X	3	30 03 00	18664.322850 R	Flow Control Frame, send next message/block
18DA20F1 X	5	21 C0 00 16 E0	18664.321310 R	Start routine by local identifier. Compare checksum. End address 7FC000h. Checksum 16E0h
18DAF120 X	3	02 71 01	18664.316450 R	Positive response on request routine results by local ID. Compare checksum
18DA20F1 X	3	02 33 01	18664.315810 R	Request routine result by local id. Compare checksum
18DAF120 X	3	02 73 01	18663.189470 R	Positive response. Checksum is correct
18DA20F1 X	2	01 20	18656.050340 R	Stop Diagnostic session (main controller)
18DAF120 X	2	01 60	18655.513500 R	Positive response
18EEFF5A X	8	06 00 20 07 02 10 00 00	18655.355810 R	Address claim from safety controller (source address 5Ah)
18EEFF13 X	8	06 00 20 07 01 10 00 00	18655.353700 R	Address claim from main controller (source address 13h)
18FECA5A X	8	10 FF E2 8F 82 30 FF FF	18655.085540 R	Application data from safety controller
18FECA13 X	8	10 FF E2 8F 82 02 FF FF	18655.084130 R	Application data from safety controller

13.6 EXAMPLE 3B: FAILURE DURING DOWNLOADING APPLICATION CODE TO THE MAIN MICRO-CONTROLLER

This example describes failure scenario of programming application code to main controller flash memory. Failure in programming application code will cause both microcontrollers to go in Bootloader mode. ECU Identification service can be used to validate if desired code is downloaded in Flash memory. Programming sequence as described in example 13.5 has to be re-executed in case of failure.

The procedure is similar for the safety controller except for the boot node id.

MSG ID	DLC	Data (hex)	Time stamp	Comment

18DA20F1 X	7	06 10 85 06 03 D0 90	18692.776860 R	Start diagnostic session. DCM = 85h (enable flash programming). Network transfer rate = 250K.
18DAF120 X	7	06 50 85 06 03 D0 90	18692.776290 R	Positive response.
18DA20F1 X	3	02 27 01	18692.775200 R	Security access - GetSeed
18DAF120 X	7	06 67 01 06 03 D0 9A	18692.771810 R	Positive response. On Security access – GetSeed. Seed = 06h 03h D0h 9Ah
18DA20F1 X	7	06 27 02 00 00 00 00	18692.771040 R	Security access – SendKey (fixed 00h 00h 00h 00h)
18DAF120 X	3	02 67 02	18692.770590 R	Positive response on Security access - SendKey
				Erase flash
18DA20F1 X	8	10 08 31 02 7E 00 00 7F	18692.768930 R	Start Routine By Local Identifier - Erase flash, start address 7E0000h
18DAF120 X	3	30 03 00	18692.768480 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 BF FF	18692.767070 R	Start Routine By Local Identifier - Erase flash, end address 7FBFFFh
18DAF120 X	3	02 71 02	18692.762210 R	Positive response on erase flash
18DA20F1 X	3	02 33 02	18692.761630 R	Request routine results by local identifier on erase flash
18DAF120 X	3	02 73 02	18687.135840 R	Positive response
				Request Download Write
18DA20F1 X	8	10 08 34 7E 00 00 00 01	18687.134940 R	Request Download - Write. Start address 7E00000h
18DAF120 X	3	30 03 00	18687.134560 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 C0 00	18687.133150 R	Request download – Write. Size in bytes 01C000h
18DAF120 X	3	02 74 38	18687.132700 R	Positive response on request download
				Transfer data – Write (56 byte block)
18DA20F1 X	8	10 39 36 56 1F 76 22 B9	18687.130910 R	Transfer data – Write. 56 bytes for transfer. 6 first bytes
18DAF120 X	3	30 03 00	18687.130460 R	Flow Control Frame, send next message/block
18DA20F1 X	8	21 C0 28 29 00 68 76 1A	18687.128540 R	Transfer data, 7 bytes
18DA20F1 X	8	22 00 7F 23 04 22 57 00	18687.127900 R	Transfer data, 7 bytes
18DA20F1 X	8	23 3F 22 57 00 3F 22 57	18687.127390 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	18687.126940 R	Flow Control Frame, send next message/block
18DA20F1 X	8	24 00 3F 22 57 00 3F 22	18687.125660 R	Transfer data. 7 bytes
18DA20F1 X	8	25 57 00 3F 22 57 00 3F	18687.125090 R	Transfer data. 7 bytes
18DA20F1 X	8	26 22 57 00 3F 22 57 00	18687.124510 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	18687.124060 R	Flow Control Frame, send next message/block
18DA20F1 X	8	27 3F 22 57 00 3F 22 57	18687.122460 R	Transfer data. 7 bytes
18DA20F1 X	3	28 00 3F	18687.122080 R	Transfer data. Last two bytes
18DAF120 X	2	01 76	18687.120800 R	Positive response on Transfer data

…

Power failure.

…..

Reset the system

Main and Safety will enter bootloader mode.
No messages will be available on CAN bus.
LED on PVED-CLS will be blinking between Green and Orange
....

Perform ECU Identification to verify download of software.

18DA20F1 X	3	02 1A 87	4237.764580 R	Request ECU identification. 87h = boot-loader SW version.
18DAF120 X	8	10 29 5A 87 42 4F 4F 54	4237.765150 R	PCI.DL = 29h (41 bytes). SI = 5Ah (pos response). IO = 87h
18DA20F1 X	3	30 03 00	4237.766240 R	Flow control frame. FS=0h. Max block size = 3
18DAF120 X	8	21 5F 43 4C 53 2D 5F 4D	4237.766820 R	Consecutive frame. Sequence no = 1
18DAF120 X	8	22 5F 52 33 37 30 5F 4B	4237.767390 R	Consecutive frame. Sequence no = 2
18DAF120 X	8	23 57 50 32 30 30 30 2D	4237.767970 R	Consecutive frame. Sequence no = 3
18DA20F1 X	3	30 03 00	4237.769310 R	Flow control frame. FS=0h. Max block size = 3
18DAF120 X	8	24 5F 30 30 30 30 30 30	4237.769890 R	Consecutive frame. Sequence no = 4.
18DAF120 X	8	25 30 30 5F 2D 72 72 72	4237.770460 R	Consecutive frame. Sequence no = 5
				The 41 bytes represents the character string:
				"BOOT_CLS-_M_R370_KWP2000-_00000000_-rrr"
18DA20F1 X	3	02 1A 99	4237.864480 R	Request ECU identification. 99h = boot-loader SW date
18DAF120 X	6	05 5A 99 07 26 13	4237.864990 R	The data represents the date "072613" (26 July 2013)
18DA20F1 X	3	02 1A 8F	4237.964450 R	Request ECU identification. 8Fh = Application SW version
18DAF120 X	8	10 29 5A 8F FF FF FF FF	4237.965020 R	
18DA20F1 X	3	30 03 00	4237.966110 R	
18DAF120 X	8	21 FF FF FF FF FF FF FF	4237.966690 R	
18DAF120 X	8	22 FF FF FF FF FF FF FF	4237.967260 R	
18DAF120 X	8	23 FF FF FF FF FF FF FF	4237.967780 R	
18DA20F1 X	3	30 03 00	4237.969250 R	
18DAF120 X	8	24 FF FF FF FF FF FF FF	4237.969820 R	The transmitted bytes represents the garbage character string:
18DAF120 X	8	25 FF FF FF FF FF FF FF	4237.970400 R	
18DA20F1 X	3	02 1A 90	4238.064610 R	Request ECU identification. 8Fh = Application date
18DAF120 X	6	05 5A 90 FF FF FF	4238.065120 R	The data represents garbage data

Garbage value in the returned Application SW version and date would imply unsuccessful download of code to flash memory.
Reprogram the ECU as described in example 13.5

13.7 EXAMPLE 4A: SUCESSFULL DOWNLOADING BOOT CODE TO THE MAIN MICRO-CONTROLLER

This example describes how downloading boot code to main controller flash memory is performed.
The procedure is similar for the safety controller except for the boot node id.

The order of programming the controllers is not important. Setting one of the controllers in boot-loader mode will automatically set the other controller is boot-loader mode as well.

MSG ID	DLC	Data (hex)	Time stamp	Comment
18DA20F1 X	7	06 10 86 06 03 D0 90	2.419680 R	Start diagnostic session. DCM = 86h (enable boot programming). Network transfer rate = 250K.
18DAF120 X	7	06 50 86 06 03 D0 90	3.022240 R	Positive response.
18DA20F1 X	3	02 27 01	3.023000 R	Security access - GetSeed
18DAF120 X	7	06 67 01 06 03 D0 5C	3.031520 R	Positive response.
18DA20F1 X	7	06 27 02 00 00 00 00	3.032540 R	Security access – SendKey (fixed 00h 00h 00h 00h)
18DAF120 X	3	02 67 02	3.032920 R	Positive response on Security access – SendKey
				Erase Flash
18DA20F1 X	8	10 08 31 02 7E C0 00 7E	3.034520 R	Start Routine By Local Identifier - Erase flash, start address 7EC000h
18DAF120 X	3	30 03 00	3.034970 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 FF FF	3.036570 R	Start Routine By Local Identifier - Erase flash, end address 7EFFFFh
18DAF120 X	3	02 71 02	3.041630 R	Positive response on erase flash
18DA20F1 X	3	02 33 02	3.042400 R	Request routine results by local identifier on erase flash
18DAF120 X	3	02 73 02	9.099290 R	Positive response
				Request Download Write
18DA20F1 X	8	10 08 34 7E C0 00 00 00	9.100250 R	Request Download - Write. Start address 7EC0000h
18DAF120 X	3	30 03 00	9.100640 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 40 00	9.102170 R	Request download – Write. Size in bytes 004000h
18DAF120 X	3	02 74 38	9.102560 R	Positive response on request download
				Transfer data – Write (56 byte block)
18DA20F1 X	8	10 39 36 00 7F 72 99 B2	9.104480 R	Transfer data – Write. 56 bytes for transfer. 6 first bytes
18DAF120 X	3	30 03 00	9.104920 R	Flow Control Frame, send next message/block
18DA20F1 X	8	21 BD AA BD A2 BD FE 1C	9.106460 R	Transfer data, 7 bytes
18DA20F1 X	8	22 76 1F 02 22 56 BF FF	9.107040 R	Transfer data, 7 bytes
18DA20F1 X	8	23 15 2B 53 76 7F 73 63	9.107610 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	9.108000 R	Flow Control Frame, send next message/block
18DA20F1 X	8	24 5C AD 28 5A 40 00 02	9.109600 R	Transfer data. 7 bytes
18DA20F1 X	8	25 00 DC 9A 56 BF 20 59	9.110170 R	Transfer data. 7 bytes
18DA20F1 X	8	26 2B 58 1E 56 76 7F 7A	9.110680 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	9.111130 R	Flow Control Frame, send next message/block
18DA20F1 X	8	27 9C 76 7F 76 AE 5C AD	9.112670 R	Transfer data. 7 bytes
18DA20F1 X	3	28 2B 41	9.113050 R	Transfer data. Last two bytes
18DAF120 X	2	01 76	9.114330 R	Positive response on Transfer data

…
Data removed to reduce example size.

...

18DA20F1 X	8	27 00 00 00 00 00 00 00	12.433820 R	
18DA20F1 X	3	28 00 00	12.434200 R	
18DAF120 X	2	01 76	12.436120 R	
18DA20F1 X	8	10 21 36 00 00 00 00 00	12.437020 R	Last 32 bytes block
18DAF120 X	3	30 03 00	12.437470 R	
18DA20F1 X	8	21 00 00 00 00 00 00 00	12.439260 R	
18DA20F1 X	8	22 00 7F 60 00 FF FF FF	12.439840 R	
18DA20F1 X	8	23 FF FF FF FF FF FF FF	12.440480 R	
18DAF120 X	3	30 03 00	12.440860 R	
18DA20F1 X	7	24 FF FF FF FF FF FF	12.442270 R	
18DAF120 X	2	01 76	12.443230 R	End of last 32 bytes block
18DA20F1 X	2	01 37	12.444000 R	Request transfer exit – Write
18DAF120 X	2	01 77	12.444440 R	Positive response on Request transfer exit
18DA20F1 X	8	10 0A 31 01 7E C0 00 7F	12.445400 R	Start routine by local identifier. Compare checksum. Start address 7EC000h
18DAF120 X	3	30 03 00	12.445850 R	Flow Control Frame, send next message/block
18DA20F1 X	5	21 00 00 A3 DC	12.447260 R	Start routine by local identifier. Compare checksum. End address 7F0000h. Checksum A3Dch
18DAF120 X	3	02 71 01	12.447710 R	Positive response on request routine results by local ID. Compare checksum
18DA20F1 X	3	02 33 01	12.449370 R	Request routine result by local id. Compare checksum
18DAF120 X	3	02 73 01	12.453080 R	Positive response. Checksum is correct

13.8 EXAMPLE 4B: FAILURE DURING DOWNLOADING BOOT CODE TO THE MAIN MICRO-CONTROLLER

⚠ This example describes failure scenario of downloading boot code to main controller flash memory. It is essential to have a stable power source and operating environment during modification of bootloader software. Failure during downloading boot code operations will lead to non-repairable damage to PVED-CLS software.

The behavior is similar for the safety controller.

MSG ID	DLC	Data (hex)	Time stamp	Comment
18DA20F1 X	7	06 10 86 06 03 D0 90	2.419680 R	Start diagnostic session. DCM = 86h (enable boot programming). Network transfer rate = 250K.
18DAF120 X	7	06 50 86 06 03 D0 90	3.022240 R	Positive response.
18DA20F1 X	3	02 27 01	3.023000 R	Security access - GetSeed
18DAF120 X	7	06 67 01 06 03 D0 5C	3.031520 R	Positive response.
18DA20F1 X	7	06 27 02 00 00 00 00	3.032540 R	Security access – SendKey (fixed 00h 00h 00h 00h)

18DAF120 X	3	02 67 02	3.032920 R	Positive response on Security access – SendKey
				Erase Flash
18DA20F1 X	8	10 08 31 02 7E C0 00 7E	3.034520 R	Start Routine By Local Identifier - Erase flash, start address 7EC000h
18DAF120 X	3	30 03 00	3.034970 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 FF FF	3.036570 R	Start Routine By Local Identifier - Erase flash, end address 7EFFFFh
18DAF120 X	3	02 71 02	3.041630 R	Positive response on erase flash
18DA20F1 X	3	02 33 02	3.042400 R	Request routine results by local identifier on erase flash
18DAF120 X	3	02 73 02	9.099290 R	Positive response
				Request Download Write
18DA20F1 X	8	10 08 34 7E C0 00 00 00	9.100250 R	Request Download - Write. Start address 7EC0000h
18DAF120 X	3	30 03 00	9.100640 R	Flow Control Frame, send next message/block
18DA20F1 X	3	21 40 00	9.102170 R	Request download – Write. Size in bytes 004000h
18DAF120 X	3	02 74 38	9.102560 R	Positive response on request download
				Transfer data – Write (56 byte block)
18DA20F1 X	8	10 39 36 00 7F 72 99 B2	9.104480 R	Transfer data – Write. 56 bytes for transfer. 6 first bytes
18DAF120 X	3	30 03 00	9.104920 R	Flow Control Frame, send next message/block
18DA20F1 X	8	21 BD AA BD A2 BD FE 1C	9.106460 R	Transfer data, 7 bytes
18DA20F1 X	8	22 76 1F 02 22 56 BF FF	9.107040 R	Transfer data, 7 bytes
18DA20F1 X	8	23 15 2B 53 76 7F 73 63	9.107610 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	9.108000 R	Flow Control Frame, send next message/block
18DA20F1 X	8	24 5C AD 28 5A 40 00 02	9.109600 R	Transfer data. 7 bytes
18DA20F1 X	8	25 00 DC 9A 56 BF 20 59	9.110170 R	Transfer data. 7 bytes
18DA20F1 X	8	26 2B 58 1E 56 76 7F 7A	9.110680 R	Transfer data. 7 bytes
18DAF120 X	3	30 03 00	9.111130 R	Flow Control Frame, send next message/block
18DA20F1 X	8	27 9C 76 7F 76 AE 5C AD	9.112670 R	Transfer data. 7 bytes
18DA20F1 X	3	28 2B 41	9.113050 R	Transfer data. Last two bytes
18DAF120 X	2	01 76	9.114330 R	Positive response on Transfer data

…

⚠ Power failure.

…..

Reset the system

Symptoms: No messages available on CAN bus and PVED-CLS does not respond to any diagnostic requests

Conclusion: Boot Loader damaged. PVED-CLS in non-repairable state.

13.9 EXAMPLE 5: MULTIPLE NODES ACCESS TO PVED-CLS

This example describes the response of PVED-CLS to request of diagnostic sessions from multiple nodes. PVED-CLS can serve requests from only one node at a time. If PVED-CLS is already in diagnostic session, no response to other nodes is provided. Active diagnostic session has to be stopped using stop diagnostic command for new nodes to be able to communicate with PVED-CLS

MSG ID	DLC	Data (hex)	Time stamp	Comment
18DA20F1 X	3	02 10 85	40.875090	Request from Node with Source Address F1h to start diagnostics
18DAF120 X	3	02 50 85	40.875530 R	Positive Response
18DA20F0 X	3	02 10 85	45.117580 T	Request from Node with Source Address F0h to start diagnostics. No Response from ECU
18DA20F0 X	3	02 10 85	51.950540 T	Request from Node with Source Address F0h to start diagnostics. No Response from ECU
18DA20F1 X	2	01 20	60.029770 R	Request from Node with Source Address F1h to stop diagnostics
18DAF120 X	2	01 60	60.656460 R	Positive Response
18EEFF5A X	8	06 00 20 07 02 10 00 00	61.777740 R	
18EEFF13 X	8	06 00 20 07 01 10 00 00	61.796810 R	
0CFF185A X	8	E8 03 00 00 FF F1 0C F8	62.087630 R	
0CFF1813 X	8	E8 03 00 00 FF F1 0C F8	62.106960 R	
0CFF1813 X	8	E8 03 00 00 FF FB 04 0A	66.609490 R	
18DA20F0 X	3	02 10 85	66.628110 T	Request from Node with Source Address F0h to start diagnostics.
18DAF020 X	3	02 50 85	67.215250 R	Positive Response
18DA20F1 X	3	02 10 85	84.536270 R	Request from Node with Source Address F1h to start diagnostics. No Response from ECU
18DA20F0 X	2	01 20	8.856850 T	Request from Node with Source Address F0h to stop diagnostics
18DAF020 X	2	01 60	99.483470 R	Positive Response
18EEFF5A X	8	06 00 20 07 02 10 00 00	00.652620 R	
18EEFF13 X	8	06 00 20 07 01 10 00 00	100.663890 R	
18FF2013 X	8	E1 FF 7F 11 FF FC A0 23	103.825610 R	
0CFF1813 X	8	E8 03 00 00 FF FA 68 C2	103.826380 R	
18DA20F1 X	3	02 10 85	103.832590 R	Request from Node with Source Address F1h to start diagnostics
18DAF120 X	3	02 50 85	104.421390 R	Positive Response

13.10 EXAMPLE 6: USING OFFICIAL MAIN BINARY (.BIOS) FILES FOR FLASHING CODE OVER KWP.

This example describes how to use the content of the official binary (.bios) file to send over KWP.

The content of the official binary files can be viewed through a hex editor or read via read binary file APIs. The below screen shot is using Notepad++ and plugin Hex-Editor.

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

APP-_CLS-_M_P196_SEHS—_11153340_-rrr_11032016_1234PM.bios

```
Address  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 56 1f 76 22 b9 c0 28 29 00 68 76 1a 00 7f 57 ac V.v"¹À().hv...W¬
00000010 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000020 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000030 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000040 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000050 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000060 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000070 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000080 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000090 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
000000a0 56 ff 00 3f 56 ff 00 3f bd ef 00 3e 56 ff 00 3f Vÿ.?Vÿ.?½ï.>Vÿ.?
000000b0 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
000000c0 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
000000d0 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
000000e0 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
000000f0 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000100 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000110 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000120 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000130 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000140 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000150 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000160 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000170 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000180 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
00000190 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
000001a0 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f 56 ff 00 3f Vÿ.?Vÿ.?Vÿ.?Vÿ.?
```

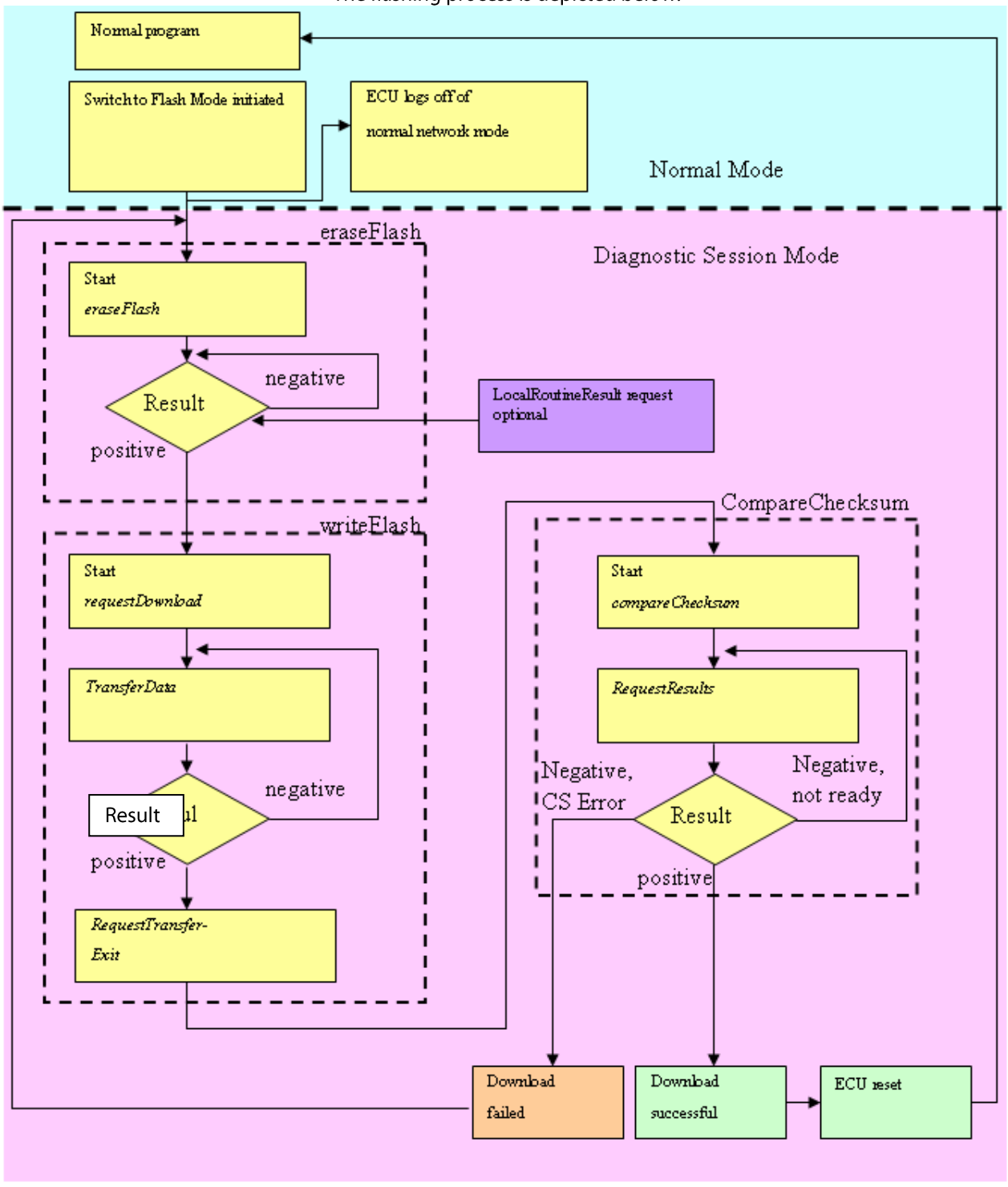
Entire file contents are not show here for readability purpose. The data available in the binary file above can be used instead of the data used in the Transfer data –Write service in example 13.5

14 Error Codes

Error Code (Hex)	Description
0x00	No error
0x7F	The basic negative response
0x11	Service not supported response
0x12	Sub function not supported response
0x22	Sequence Error
0x80	Request Length Error response
0x78	Response Pending Negative response
0x35	The Security key provided is not valid
0x33	This session is protected and a valid key has not been provided
0xFA	The format of the received message was invalid
0x41	The target type or data identifier are wrong
0xFA	Invalid length of download data
0x23	The routine is not complete (eg. Erase Flash)
0xFC	The Transfer Data Framer format is incorrect
0xFE	Incorrect blockcount on data transfer
0x79	Wrong length of downloaded data
0xFA	Wrong number of bytes received
0xFA	Invalid Checksum on received download
0x22	Incorrect Condition
0x31	Invalid Baud rate
0x43	Cannot download data further
0xFE	Error during programming
0xFD	Error during erasing
0x81	Error for Timeout
0x82	Error for Invalid application

15 Flashing process

The flashing process is depicted below.



16 PLUS+1 Service tool protocol Support

The PVED-CLS boot-loader is also compatible with the PLUS+1 GUIDE KWP2000 protocol variant which is used by the PLUS+1 service tool. This protocol is not described in this document. Contact your local Danfoss PAE for further information on the PLUS+1 service tool.