工程研究院
内部资料

北汽新能源
BAIC EU

整车各零部件电控单元 基于 UDS 协议的 Bootloader 刷新规范

文件编号	：
编制	：[illegible] 2016.1.11
校对	：[illegible] 2016.1.11.
审核	：[illegible] 2016.1.12
会签	：[illegible]
批准	：代康伟

[illegible] 1.26

北京新能源汽车股份有限公司

2015 年 12 月

1 Document Information

1.1 History Records

Author	Date	Version	Description	Remarks
BAIC EV Technical Centre Software Develop Branch	2014-01-29	1.0(Draft)	Bootloader Specification	First Release
BAIC EV Technical Centre Software Develop Branch	2014-07-29	2.0(Draft)	Bootloader Specification	
BAIC EV Technical Centre Software Develop Branch	2015-03-07	3.0(Draft)	Bootloader Specification	
BAIC EV Technical Centre Software Develop Branch	2015-12-31	4.0(Draft)	Bootloader Specification	1.Add 1.2 Documents ' ISO 15765-3'; —P3 2. Modify 4.6 description; —P18 3. Modify 5.2 STmin '20ms->1ms'; —P20 4. Update 6.3.3.4.2 NRC of 27 service 'Add 0x37'; —P35 5.Add the programming tools constraint 'PCAN'; —P58 6. Add the number of attempts 'three times'; —P59 7.Modify Non VCU control program process 'Reference 3)、1)'; —P60

Table 1-1 History of the Document

1.2 Reference Documents

No.	Source	Title	Version
[1]	BAIC EV	BAIC EV Diagnostic Specification	1.1
[2]	ISO	ISO14229-1 Road vehicles - Unified Diagnostic Services	

		(UDS)Part 1	
[3]	ISO	ISO 15765-2: Road vehicles – Diagnostics on CAN – Part 2	
[4]	ISO	ISO 15765-3: Road vehicles – Diagnostics on CAN – Part 3	
[5]	BAIC EV	BAIC EV internal document – CAN ID assignment	

Table 1-2 Reference Documents

1.3 Document Conventions

In this document, the following terminology applies:

- "Shall" expresses an obligatory/mandatory requirement.
- "Should" expresses a recommendation or an advice.
- "Must" expresses a legal or normative requirement.
- "Will" expresses a precautionary consideration or an additional / optional feature.
- "May" expresses a permitted practice / method, not to be considered as a requirement.

1.4 Glossary

Term	Description
Client	External diagnostic tool (tester), that transmits service requests to ECUs
Logical Block	Reserved portion of target memory, where application data can be downloaded (like hard disk partition).
Logical Block Table	The target memory is split up in several Logical Blocks. The LogicalBlocktable acts like a file system partition table. If application data is to be downloaded, the Bootloader checks, if there is a valid entry in the LogicalBlocktable for the download.
Server	ECU that responds to diagnostic service request for an externaldiagnostic tool.

1.5 Abbreviations

Term	Description
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read Only Memory
FBL	Flash Boot Loader
DTC	Diagnostic Trouble Code
CAN	Controller Area Network

CRC	Cyclic Redundancy Check
MCU	Micro Controller Unit
TP	Transport Protocol
Tpup	Time for power up

目录

2 Introduction

2.1 Scope of this Document

This document specifies the technical requirements for flash reprogramming based on UDS. A reprogramming process including standardized interfaces and communication sequences is needed to be able to support ECU software updates during the development phase and in-service of automotive ECUs. This specification provides the requirements for reprogramming, the architecture and the download process flow to support flash reprogramming of an ECU.

This document focuses on CAN based systems only. It does not replace the existing normative documents regarding diagnostics, but it adds additional requirements and restrictions to the contents of the standards. In case of conflicts this document takes priority over the normative documents.

The content of this specification is mandatory for all reprogrammable and non-reprogrammable ECUs. Any deviation requires the approval of BAIC EV and shall be documented within the corresponding ECU diagnostic specification.

2.2 Target Group

This document is intended for development engineers, suppliers of diagnostic equipment and ECU suppliers who are engaged in reprogramming and software updating of ECUs and those who have to equip their ECU with flash reprogramming capabilities.

2.3 Fundamentals

A reprogrammable ECU contains two executable software packages, the real ECU application software and the bootloader software. During normal ECU operation, the application software is executed. The bootloader software is active only if it is started by the application software in order to download a modified version of the application software, or if no application software is available on the system.

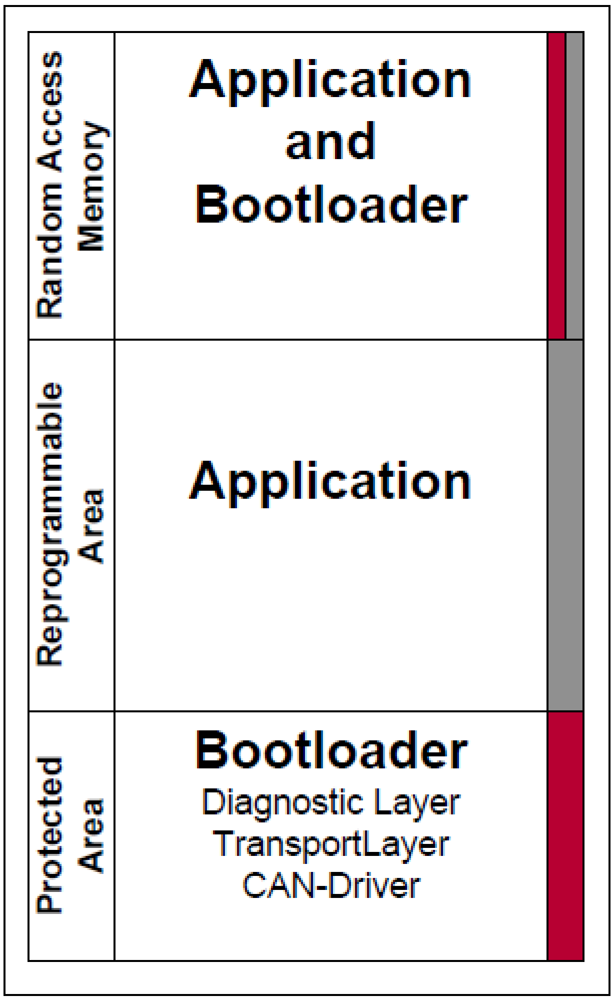


Figure 2-1 Symbolic memory map of a reprogrammable ECU

Application software and bootloader software occupy a dedicated area of flash memory as depicted in the figure above. Since either the bootloader or the application is executed, the RAM memory of the system can be fully used by both software packages.

The bootloader software uses the diagnostic services of UDS as protocol for download communication. Therefore, the bootloader has got a communication stack of CAN driver, transport layer and a subset of the UDS protocol layer.

3 Flash EEPROM Reprogramming Requirements

3.1 Requirements for non-reprogrammable ECUs

In order to operate reprogrammable and non-reprogrammable ECUs together in one network, non-reprogrammable ECUs shall support the diagnostic services specified in6.3.2 except the DiagnosticSessionControl –programmingSession request.

If a non-reprogammable ECU receives a DiagnosticSessionControl –programmingSession ($10 $02) it shall send the negative response code $12(subFunctionNotSupported).

3.2 General Requirements

The system shall be able to enter the bootloader on request from normal operating mode or if no ECU application software is available.

The system shall be reprogrammable after a failed or interrupted reprogramming, after a timeout or a reset occurred, and with partly erased flash memory or invalid application.

The system shall execute boot code if the application code is missing, invalid or corrupt.

A system executing boot code must not disturb normal communication on the network.

All ECUs connected to the network shall disable their normal message transmission when they are not reprogrammed and after they received a disable normal message transmission request. Normal communication shall be enabled on all ECUs after an explicit request to enable normal communication, after return to the default session and after a PowerOn/Reset.

All ECUs connected to the network shall disable their DTC settings after they received a request to disable fault code setting. Fault code setting shall be enabled on all ECUs after an explicit request to enable fault code setting, after return to the default session and after a PowerOn/Reset.

After all download data has been transferred, the integrity of the programmed data shall be ensured by a verification routine. Therefore, an explicit check value shall be provided to be compared against the value calculated on the ECU. For further information, refer to 4.6.2.
If a watchdog is used on the ECU, it shall be served during the complete flash reprogramming sequence.
A programming dependency check shall be performed after download if more than one module can be reprogrammed on the system. For further information, refer to 6.2.2.7and 6.3.3.7.4.

3.3 Resource Requirements

The flash memory consumption of the bootloader must be kept to minimum since the available flash memory of an ECU is shared by application and bootloader. The available ECU RAM memory can be fully occupied by the bootloader and is also fully available for the application.

In order to reduce power consumption, the ECU may enter its sleep mode when it executes boot code in idle mode (Default Session) for more than 600 seconds.

3.4 Security Requirements

Before a reprogramming procedure is started, the security access service shall be passed successfully. Additional requirements of security access are specified in Annex A.

Critical parts of the flash driver code must not be stored on the ECU without encoded.

The boot software shall be stored in a protected memory area (software and/or hardware protection).

3.5 Download Of Flash Data

3.5.1 Addressing

The addressing scheme for download data is based on a linear address space. Depending on the requirements of the underlying micro-controller platform, linear addresses must be transformed into physical addresses by the bootloader whenever flash memory is accessed for reading, writing and erasing. The conversion of linear addresses into physical ones shall be done individually according to the hardware-specific conventions. It is also possible that no conversion is necessary. That way, it is possible to support banked flash memory devices, EEPROM and external flash memory devices in one ECU, even for the ECU which contain several Micro-Controllers.

3.5.2 Erasing Of Flash Memory

Flash devices require that flash memory is erased before it is reprogrammed. Any flash cell must not be reprogrammed without a previous erase procedure in order to achieve the full data retention time of the device.

3.5.3 Ascending Address Order

All download data must be transferred to the ECU in ascending address order as depicted in Figure 4-3.This requirement makes sure that there is no multiple subsequent write access to flash cells without erasing.

3.5.4 Flash Programming Conditions

Before the flash reprogramming procedure is started, programming conditions must be checked. The ECU that shall be reprogrammed might be in a condition that does not permit ECU reprogramming. In this case, a reprogramming must be rejected. The check for flash programming conditions shall be performed when the ECU receives the RoutineControl($31) – “checkProgrammingPreconditions($02$03)” service request.

3.5.5 Programming Counter

There may be some reasons to limit the number of programming event for an ECU, e.g. because a flash device only permits a certain number of erase and programming cycles.

The bootloader software shall count the number of programming event for each logical block. When RoutineControl $31(start eraseMemory) is requested, bootloader should check the programming counter of the related logic block referring to the request of erasing address and size. If the programming counter exceeds a predefined value, the current programming must be denied by sending a negative response upon the service request RoutineControl $31(start eraseMemory).If erasing successfully, the programming counter of each logic block should only be increased by one.

The maximum number of reprogramming Counter depends on the applied flash device and the limitations defined by the OEM and shall be configured for each ECU individually. If the maximum number of programming Counter is set to the value of 0, programming times are unlimited.

4 Architectural Requirements

4.1 Functional Overview

The bootloader software is subdivided into several functional blocks as depicted in Figure4-1. It is stored in the protected memory area of the ECU so that the bootloader can be activated at any time. The ECU can be reprogrammed despite of potential error conditions.

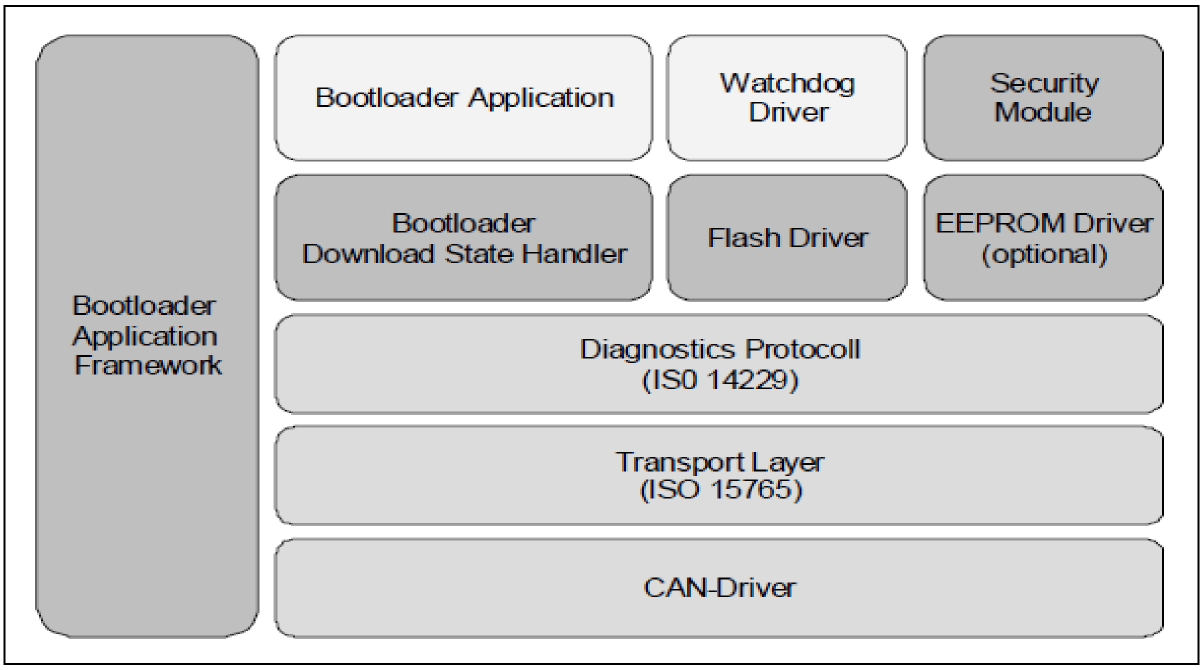

Bootloader Application Framework
Bootloader Application
Watchdog Driver
Security Module
Bootloader Download State Handler
Flash Driver
EEPROM Driver (optional)
Diagnostics Protocoll (IS0 14229)
Transport Layer (ISO 15765)
CAN-Driver

Figure 4-1 Functional blocks of the bootloader

The functionality of the shown blocks is described in the table below.

Functional Block	Description
Bootloader Application Framework	The bootloader application framework provides the boot procedure and the time base handling for the bootloader components.
Bootloader Application	This module permits supplier-specific code in order to support system specific implementations.
Watchdog Driver	The watchdog driver implements an ECU-specific code to serve the watchdog(s) of the system
Security Module	The security module provides implementations required for the security access service and the verification of the download.
Bootloader Download State Handler	This component of the bootloader is closely related to the diagnostic layer and controls the states of the download sequence and the flash

	programming
Flash Driver	The flash driver provides routines for flash erasing and programming.
EEPROM-Driver	The EEPROM driver provides routines for EEPROM erasing and programming.
Diagnostic Protocol	This module provides all diagnostic services needed for the download procedure.
Transport Protocol	ISO-15765-2 -compliant transport layer to support segmented transmission of diagnosis/ download data.
CAN-Driver	Hardware specific CAN driver

Table 4-1 Description of the functional blocks

4.2 Flash Memory Partitioning

4.2.1 Physical Flash Sectors

The bootloader shall take into account that the underlying flash technology determines the minimum number of bytes that must be erased all at once. This size is called a physical flash sector. The memory area of a flash device usually is divided into several flash sectors.

4.2.2 Physical Flash Pages

The bootloader shall take into account that the underlying flash technology determines the minimum number of bytes that must be programmed all at once. This size is called a physical flash page.

4.2.3 Logical Blocks

The flash memory space of an ECU can be partitioned into logical blocks. One logical block can be reprogrammed independently from other blocks. The basic motivation for flash partitioning is to avoid reprogramming of the complete flash memory when only a certain part of the application software or application data has changed.

The partition that is occupied by the bootloader is called boot block and cannot be reprogrammed. The contents of a reprogrammable logical block can be determined by the ECU supplier and can consist of code, data or both. The supplier can decide if and how the application code and data is subdivided into units using logical blocks.

Logical blocks must be aligned to physical flash sectors. That means that a block shall start at a physical flash sector and shall comprise an integer number of flash sectors. This requirement makes sure that each logical block can be erased and reprogrammed individually without affecting other blocks. For the same reason, logical blocks must not overlap, and they must not share one common physical flash sector.

Logical blocks must not be nested.

Logical block are handled by the so called “logical block header”, which located at the start address of each logical block. Every logical block header consists of the block number, validation CRC value, the start address and the length information of the concerned block.

In one ECU at least one logical block must be available.

Each block is identified by an individual block number. This block number is used to access the administrative data of the block. Block numbering starts at “0”.

For each logical block, a set of administrative data, also called meta data, shall be maintained by the bootloader software. Logical block header is a part of meta data. Detail information of meta data are specified in 7.2 Meta Data Table.

4.2.4 Configuration Examples

To illustrate the usage of logical blocks, two configuration examples with logical blocks are discussed here.

Configuration 1 of Figure 4-2 shows the ECU flash memory with the bootloader code in the protected memory area and one logical block that contains all application code and data. The complete logical block 0 can only be erased and reprogrammed at once.

Configuration 2 of Figure 4-2 shows the ECU flash memory with the bootloader code in the protected memory as in configuration 1, but the reprogrammable area consists of several logical blocks. In this example, block 1 contains pure application data so that this block can be reprogrammed individually and without changing the contents of other blocks if data in this block has changed.

Administrative effort is kept to a minimum when a logical block configuration like configuration 1 is applied, because one logical block requires just one set of identification data. When several logical blocks are used, an individual set of identification data shall be administrated by the flash reprogramming process.

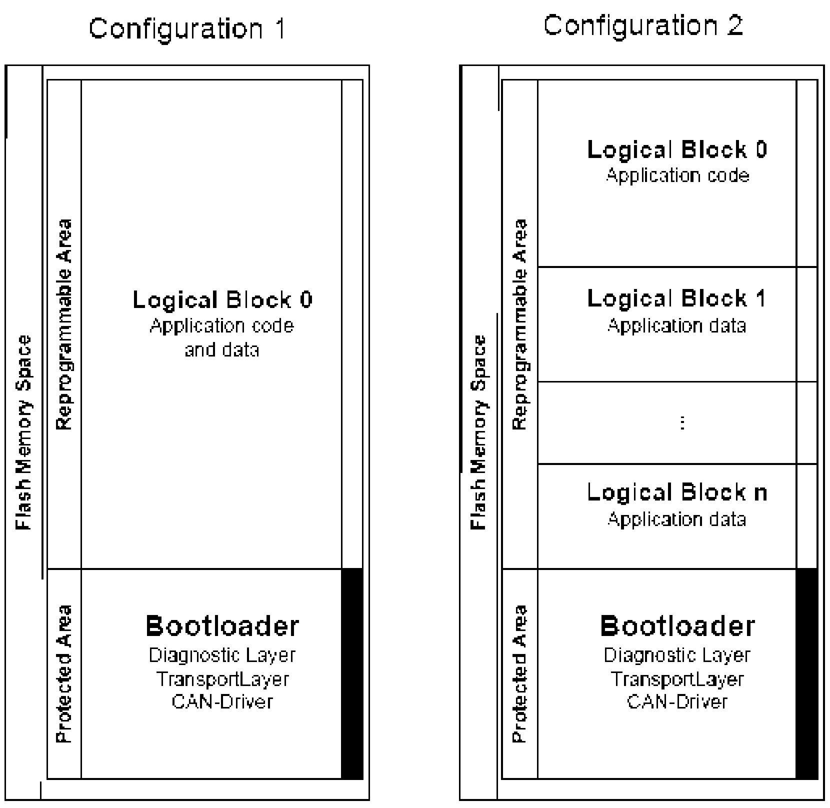


Figure 4-2 Configuration of logical blocks

4.2.5 Segmentation

The contents of logical blocks may consist of contiguous and non-contiguous data. The code of a logical block can be segmented e.g. because the linker generates different segments for the application code-, text-, const- and may be several other segments.

A segment must reside inside of the address range of the current logical block.

A segment must be aligned to integer multiples of flash pages according to the requirements of the underlying flash device.

One segment is downloaded by a sequence of the diagnostic service requestsRequestDownload – TransferData –RequestTransferExit(see section 6.2.2.5). This is repeated for all segments of the logical block.

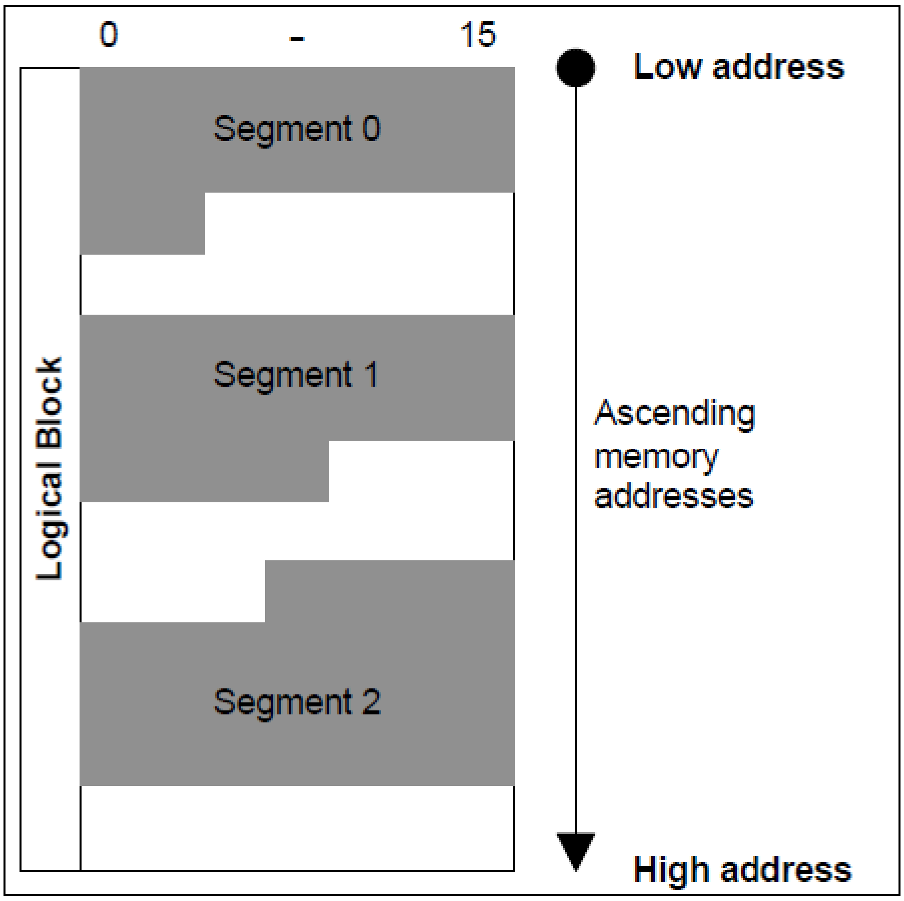


Figure 4-3 Segmentation of logical block data

4.3 ECU Startup Sequence

After PowerOn/Reset, boot code is executed first.The bootloader performs some basic initializations and then checks if an external reprogramming request message ($10 $03) is to be send or if the external reprogramming request flag was set by application within 20ms Tpup. These reprogramming checks should be done within 20ms. Boot should clear the external reprogramming request flag if it was set .If both upon requests occurs, bootloader should step into external session or programming session. In this case, Flash-bootloader code is executed furthermore.

If no reprogramming request is present or even if the request is present but without 20ms from ECU PowerOn, the status of the application will be checked. If the application is valid, the bootloader starts the application. If the application is not valid, bootloader code is executed.

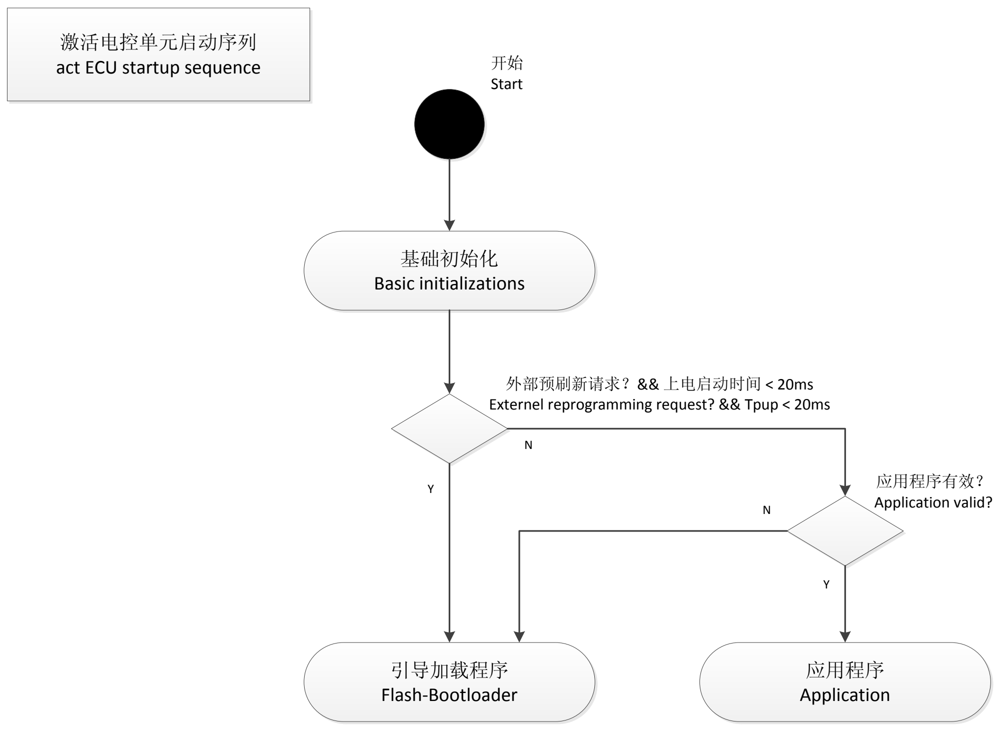


Figure 4-4 ECU Startup Sequence

4.4 Flash Driver

The flash driver is a hardware-dependent module that provides the functionality of erasing and programming the ECU flash memory.

The flash memory contents must be secured against unintended erasure and overwriting. Therefore, a software interlock mechanism is implemented in the bootloader code that stores critical code outside of the ECU memory. The complete flash driver code or critical parts are not stored in the ECU flash memory but are downloaded into an ECU RAM buffer during the download procedure.

After the download is completed the flash driver code shall be explicitly removed from the RAM buffer before the ECU returns to normal operation mode.

Downloading the flash driver into a RAM buffer when it is needed is an additional advantage, because flash memory resources can be saved this way. Additionally, on most micro-controller platforms, flash memory cannot be erased or programmed by code stored in flash memory or at least in the same flash bank.

At least two functions should be protected, the erase and write operation.

4.5 Watchdog Support

On most ECUs at least one watchdog is to be served in the bootloader. Therefore, the bootloader shall provide a watchdog support to serve either normal or windowed watchdog devices. The watchdog device can be either an internal one on the micro-controller or an external one.

Some watchdog devices permit to switch between a “short” and a “long” operating mode.

Sometimes these modes are also called “fast” and “slow”.

The bootloader shall support these operation modes by providing ECU-specific functions so that the required operations can be performed.

4.6 Security Access

Flash reprogramming shall be secured against unauthorized download attempts. Therefore, the security access procedure must be passed successfully before further reprogramming steps are unlocked.

In the programming session, the seed level $11 and the key level $12 is used via the securityAccess ($27) service.

The security access algorithm is provided by BAIC EV.

4.7 Integrity Verification

Every logical block is secured with a CRC32 value. After a download, the bootloader must ensure that all data bytes of the current block have been transferred and written correctly. Therefore, a CRC32 algorithm as verification routine is applied. The verification routine is called by the bootloader when the RoutineControl service request with checkRoutine routine identifier is received and calculates the CRC32 of the downloaded data bytes. After the verification routine called, bootloader should compare the resulting value with the reference check value that is transmitted to the ECU in the service request message. The verification routine is applied for the flash driver and for the application download. But for the application, if upon check is passed, bootloader should check every logical block according to each logical block header. The header of each logic block of programming files is made by IDE or external tools and APP developer should make sure that the header can validate the whole logic block.

The header of logical block is depicted in Figure4-7

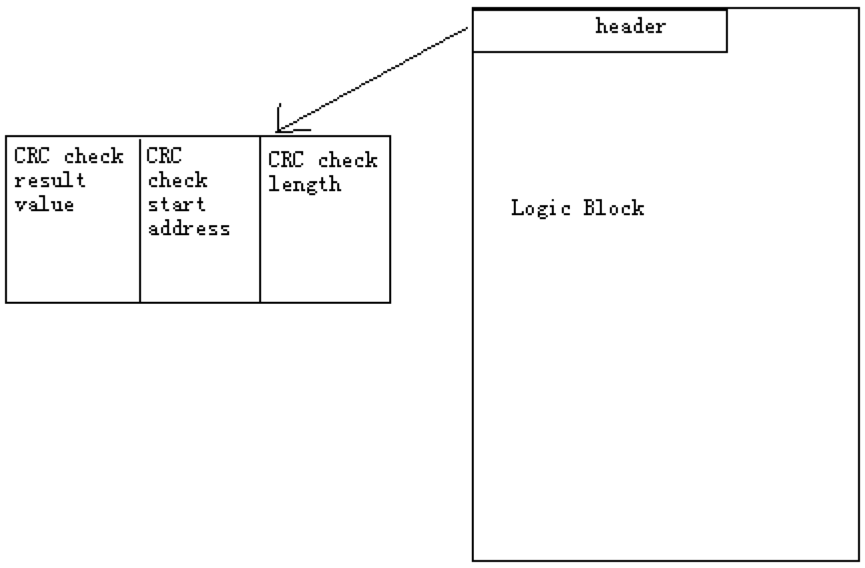


Figure4-7

For the CRC32 computation, the following generator polynomial is used:

G(x) =x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

5 Network Layer Parameters

5.1 CAN Communication Parameters

The addressing scheme and the CAN Identifier assignment are available in [5].

DLC of the download CAN-message is fixed to 8 bytes. ECU should ignore any other value of DLC of the diagnosis download CAN-message.

The Padding data is $AA, if the data is less than 8 bytes.

5.2 Transport Layer Parameters

The transport layer implementation for flash download shall be based on ISO 15765- 2 .

The transport layer timing parameters (N_As, N_Br,Blocksize,STmin etc.) shall be set as specified as listed in the table below.

Transport layer timing requirements	
N_As Timeout value	70ms
N_Ar Timeout value	70ms
N_Bs Timeout value	150ms
N_Br Timeout value	N/A
N_Cs Timeout value	N/A
N_Cr Timeout value	150ms

Flow Control Mechanism Requirement Values	
BlockSize Fixed value	8
STmin Fixed value	1ms

Diagnostics layer timing requirements	
P2 parament value	P2 = 50ms
P2* parament value	P2* = 2000ms
Session timeout	S3 = 5000ms

6 Diagnostic Communication

Flash reprogramming is controlled by a subset of the diagnostic communication protocol.

This chapter specifies the requirements for diagnostic communication like session management, the reprogramming sequence and the relevant diagnostic services.

6.1 Diagnostic Session Management

In connection with the flash reprogramming, there are three different diagnostic sessions used, the default session, the extended diagnostic session and the programming session. After a PowerOn/Reset occurred, the ECU is started as described in 4.3.

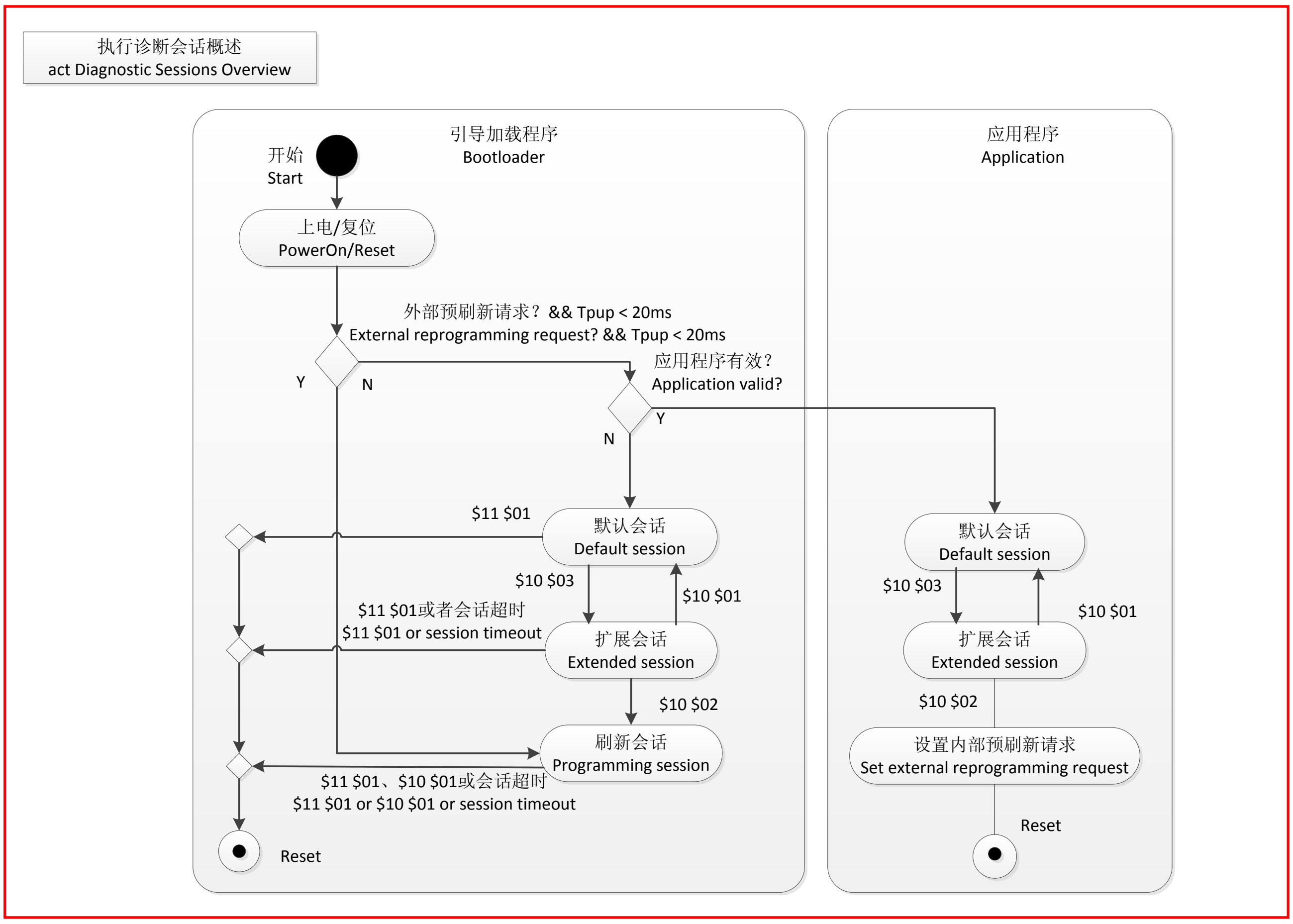


Figure 6-1 Diagnostic Session Overview

If no external reprogramming request is present, the bootloader determines if a valid application is available. If a valid application is detected, the application is started in default session.

If the application is invalid, the ECU continues executing bootloader code in the default session and waits for requests to switch to extended session and to programming session for ECU reprogramming.

The extended session of the bootloader can be left by a default session request so that the default session of the bootloader is entered.

If the ECU is executing its operative application software in the extended diagnostic session, a DiagnosticSessionControl service request with programmingSession sub parameter leads to the activation of the bootloader. Therefore, the application shall set the external reprogramming request flag and shall issue a Reset. Upon the Reset, the ECU is starting-up as specified in Figure 6-1. When the bootloader is started by the application with $10 $02, the programming session of the bootloader is active.

The programming session will be left by a session timeout, by an ECUReset or a defaultSession service request.

The session transition request from programming session to the extended session within the Bootloader shall not be supported.

6.2 Flash Reprogramming Sequence

The flash reprogramming sequence is divided into three steps, the pre-programming step, the server programming step and the post-programming step. These three steps are described in detail in the following sections. In the sequences depicted in Figure 6-2, Figure 6-3, and Figure 6-4, optional steps are shown on the right side, similar to the graphics in the ISO document.

Note: the pre-programming step and the post-programming step are only supported in application .if these services of them are sent while ECU is running in bootloader , no responds or negative responds should be sent .

6.2.1 Pre-Programming Step

The pre-programming step is used to prepare the network for the programming procedure.

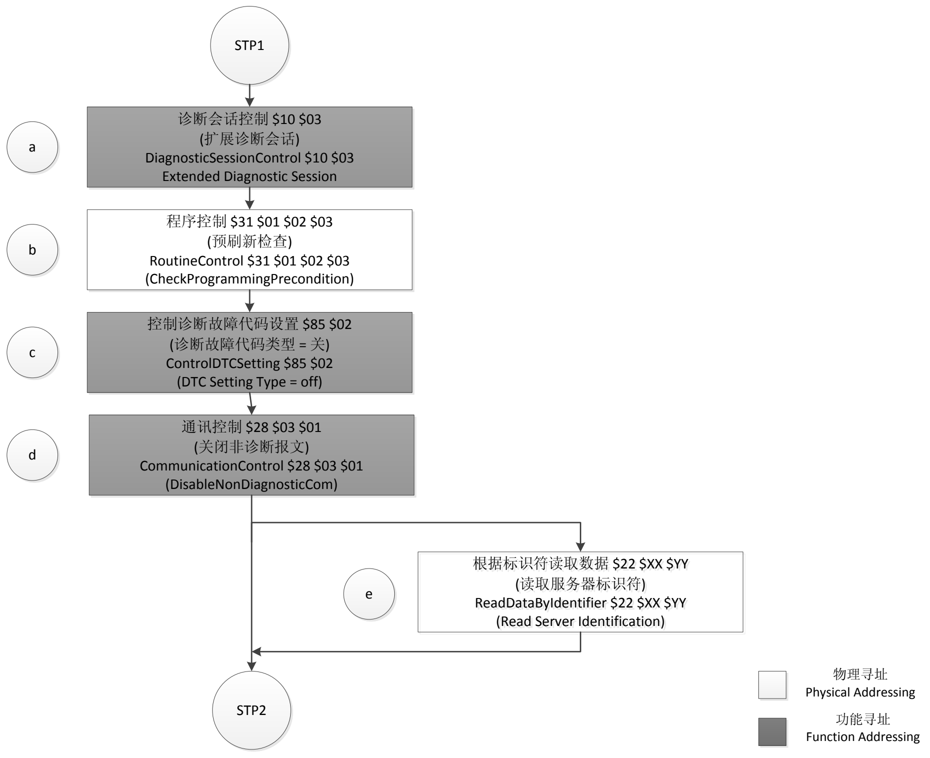

STP1
a
诊断会话控制 $10 $03
(扩展诊断会话)
DiagnosticSessionControl $10 $03
Extended Diagnostic Session
b
程序控制 $31 $01 $02 $03
(预刷新检查)
RoutineControl $31 $01 $02 $03
(CheckProgrammingPrecondition)
c
控制诊断故障代码设置 $85 $02
(诊断故障代码类型 = 关)
ControlDTCSetting $85 $02
(DTC Setting Type = off)
d
通讯控制 $28 $03 $01
(关闭非诊断报文)
CommunicationControl $28 $03 $01
(DisableNonDiagnosticCom)
e
根据标识符读取数据 $22 $XX $YY
(读取服务器标识符)
ReadDataByIdentifier $22 $XX $YY
(Read Server Identification)
STP2
物理寻址
Physical Addressing
功能寻址
Function Addressing

Figure 6-2 Pre-programming Step

6.2.1.1 (a) DiagnosticSessionControl $10 $03

Before an ECU can be reprogrammed, the setting of DTCs and the normal communication in the network shall be disabled. Therefore, a non-default diagnostic session shall be started on all ECUs connected to the network. Therefore, this request uses functional addressing. With this service request the external reprogramming tool shall start transmitting TesterPresent messages in order to keep all ECU of the network in the extended diagnostic session.

6.2.1.2 (b)RoutineControl $31 $01 $02 $03

The check of programming preconditions is required to make sure that the system is in a safe state so that the reprogramming procedure can be carried out. This diagnostic service must be responded.

6.2.1.3 (c)ControlDTCSetting $85 $02

With a ControlDTCSetting service request with DTCSettingType set to “off” to disable detection and storage of DTCs by an ECU during the reprogramming procedure. This request uses functional addressing to disable fault code setting on all ECUs of the network.

6.2.1.4 (d)CommunicationControl $28 $03 $01

With a CommunicationControl service request, all ECUs connected to the network disable transmission of normal, non-diagnostic messages. Having normal message transmission disabled, an ECU receives and processes incoming messages but does not send a response. By disabling normal message transmission, the full bandwidth of the bus is available for the download and the download is not disturbed by non-diagnostic messages.

6.2.1.5 (e)ReadDataByIdentifier $22 $xx $yy

The ReadDataByIdentifier service is optional and used to acquire identification information from the ECU that is be reprogrammed. ECU identification information is applied by the flash process infrastructure to determine the suitable software update that shall be programmed and to document the download event.

6.2.2 Server Programming Step

The server programming step is used to program one or several logical blocks. All service requests of this step use physical addressing, so that it is possible to program multiple nodes in parallel. Several logical blocks can be programmed without terminating the server programming step.

The download of one logical block is initiated by theWriteDataByIdentifier service request and is terminated by the successful verification of the download by the RoutineControl with the checkRoutine routine identifier.

If an error occurs during the server programming step, the full sequence shall be performed again.

If a timeout occurs while processing the transferData service ($36), the block sequence counter shall be used for improved error handling as specified in [2].

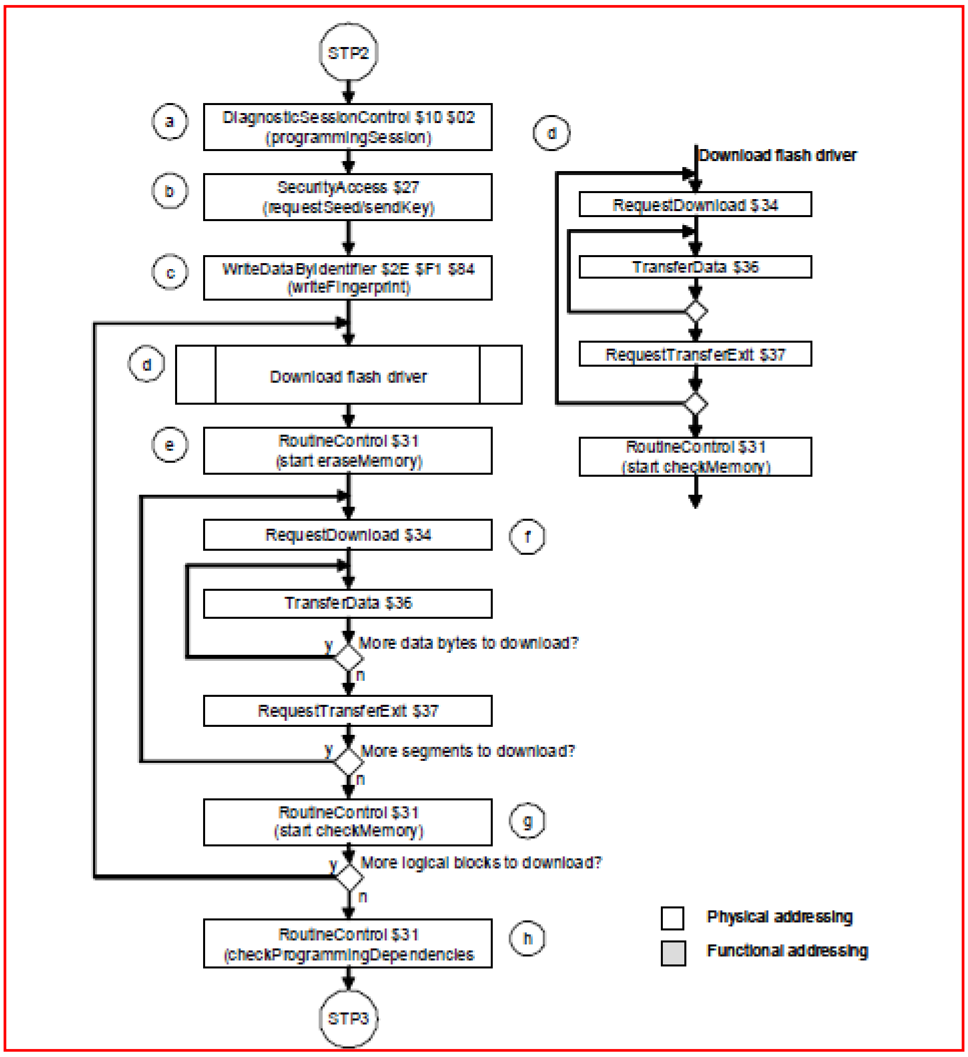

STP2
a
DiagnosticSessionControl $10 $02 (programmingSession)
b
SecurityAccess $27 (requestSeed/sendKey)
c
WriteDataByIdentifier $2E $F1 $84 (writeFingerprint)
d
Download flash driver
e
RoutineControl $31 (start eraseMemory)
RequestDownload $34
f
TransferData $36
More data bytes to download?
y
n
RequestTransferExit $37
More segments to download?
RoutineControl $31 (start checkMemory)
g
More logical blocks to download?
RoutineControl $31 (checkProgrammingDependencies
h
STP3
d
Download flash driver
RequestDownload $34
TransferData $36
RequestTransferExit $37
RoutineControl $31 (start checkMemory)
Physical addressing
Functional addressing

Figure 6-3 Programming Step

6.2.2.1 (a)DiagnosticSessionControl $10 $02

The download procedure is started in the ECU with a DiagnosticSessionControl service request with a sub function parameter set to programmingSession. In an ECU, that is executing its operative application software, this service request leads to a transition from application software to bootloader software. While the reprogramming session is active, reprogramming specific service requests are supported.

6.2.2.2 (b)SecurityAccess $27 $11/$12

Security access shall be granted to the tester by the ECU before flash reprogramming. To unlock an ECU, the tester must request the seed from the ECU first. Then, both the tester and the ECU calculate the key and the tester transmits it to the ECU. The ECU compares its key value with the one received from the tester and if the values match, security access will be granted to the tester.

6.2.2.3 (c)WriteDataByIdentifier $2E $F1 $84

Prior to any flash access, the fingerprint of the tester shall be stored on the ECU's non-volatile memory. The fingerprint is transferred to the ECU by a WriteDataByIdentifierservice request.

6.2.2.4 (e) RoutineControl– Start eraseMemory $31 $01 $FF $00

The RoutineControl service requested with parameter $01 $FF $00 erases the requested logical block. Before the erase routine of the flash driver is called, the validity status of the logical block in question must be set to invalid. This prevents accidental execution of the application if the flash process does not terminate successfully. Additionally, the fingerprint that has been received with a previous WriteDataByIdentifier service request shall be stored in non-volatile memory before the logical block is erased.

6.2.2.5 (f) Download Process $34,$36,$37

All segments of a logical block are downloaded to the ECU by a sequence of the servicesRequestDownload, TransferData and RequestTransferExit. The download of one segment that consists of a contiguous set of data bytes is started with a RequestDownload service request. The RequestDownload service informs the bootloader about the start address in memory and the length of the segment. After RequestDownload, all data bytes of the segment are transferred by one or more subsequent TransferData requests. After all segment bytes are transferred, the download of a segment is terminated with a RequestTransferExit request. This sequence can be repeated for several times depending on the number of segments in a logical block.

6.2.2.6 (g)RoutineControl – Start checkRoutine $31 $01 $02 $02

After all bytes of a logical block are transferred to the ECU, the verification routine is started to ensure integrity (and optionally authenticity) of the downloaded data.

If there is another logical block to be downloaded, the server programming sequence can be continued with service request.

6.2.2.7 (h)RoutineControl – Start checkProgrammingDependencies $31 $01 $FF $01

To make sure that the reprogrammed logical blocks are consistent and compatible, i.e. that all blocks e.g. use compatible interfaces and formats, this service request is sent to theECU after the download of the last logical block. With this service, checks can be implemented to make sure that e.g. the application data is compatible to the application software or that the interfaces of two software modules match.

Only if this check is successful, the application can be started.

If only one logical block is used on an ECU, the routineStatusRecord of the positive response is always $00 – correctResult.(see also 6.3.3.7.4)

6.2.3 Post-Programming Step

The post programming step is performed to conclude the programming event after the programming step of a reprogrammed ECU is finished.

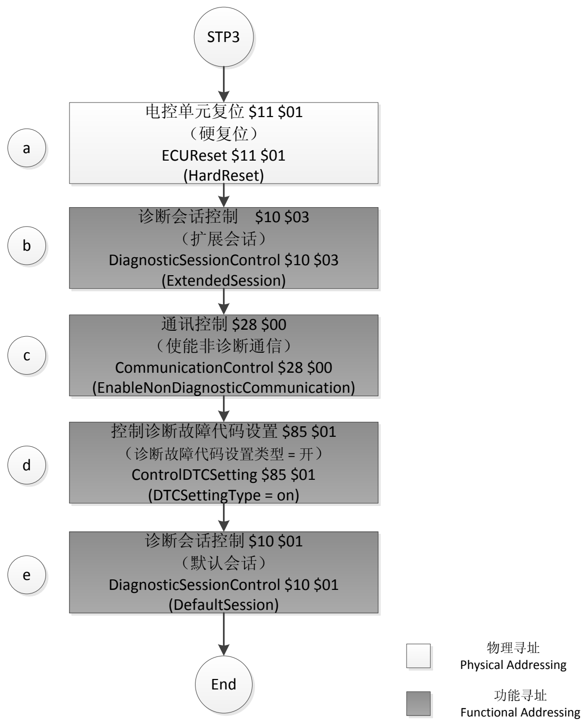


Figure 6-4 Post-Programming Step

Author		Data	2015-12-31	Version	4.0
Description	BAIC EV Bootloader Specification Based on UDS			Page	27/63

6.2.3.1 (a)ECUReset $11 $01 (Physical addressing)

The flash reprogramming process is terminated by an ECUReset service request in order to return to normal ECU operation.

The flash driver code must be removed completely from its RAM buffer to avoid accidentalactivation of the code that might end up in unintended erase or program operations.

6.2.3.2 (b)DiagnosticSessionControl $10 $03

After the EcuReset request (a), the currently reprogrammed ECU is in the default diagnostic session. In order to perform the next diagnostic services of the post programming step that require the extended session, a DiagnosticSessionControlExtendedSession request shall be transmitted using either physical or functional Addressing.

6.2.3.3 ©CommunicationControl $28 $00 $01

A functional message requesting service CommunicationControl – enableNonDiagnosticCommunication shall be transmitted to all ECUs connected to the network to return to normal communication.

6.2.3.4 (d)ControlDTCSetting $85 $01

After the download has been performed, the setting of DTCs must be enabled again for all ECUs connected to the network by the functionally addressed request ControlDTCSettingwith sub parameter DTCSetting'Type set to on.

6.2.3.5 (e)DiagnosticSessionControl $10 $01

Finally, all ECUs of the network return to default session.

6.3 Diagnostic Services Overview

6.3.1 Diagnostic Services in Bootloader Software

This chapter gives an overview of the UDS diagnostic services subset that is required in the boot software. The following table shows the required service Ids and the supported sub function parameters. Additionally, the table shows if a certain service request is addressed physically, functionally or both. The last three columns indicate if a service is supported in the default-, programming or extended diagnostic session.

X: Service request supported for phys. / funct. Request; request supported in session XY

-: Service request not supported for phys. / funct. Request; request not supported in session XY

Diagnostic Service Description	Hex vale	Phys.Req.	Funct.Req.	Supported in session		
Subfunction Parameter				**$01**	**$02**	**$03**
DiagnosticSessionControl	10					
defaultSession	10 01	x	x	x	x	x
programmingSession	10 02	x	-	-	x	x
extendedDiagnosticSession	10 03	x	x	x	-	x
ECUReset	11					
Power On Reset	11 01	x	x	x	x	x
ReadDataByIdentifier	22					
Data Identifier	22 xx yy	x	-	x	x	x
SecurityAccess	27					
Request Seed(for reprogramming)	27 11	x	-	-	x	-
Send Key(for reprogramming)	27 12	x	-	-	x	-
CommunicationControl	28					
disableRxAndTx	28 03 01	x	x	-	-	x
enableRxAndTx	28 00 01	x	x	-	-	x
WriteDataByIdentifier	2E	X	-	-	X	-
applicationSoftwareFingerprintDataIdentifier	2E F1 84					
RoutineControl	31					
CheckProgrammingPreconditions	31 01 02 03	x	-	-	-	x
Start Erase Memory	31 01 FF 00	x	-	-	x	-
Start Check Routine	31 01 02 02	x	-	-	x	-
CheckProgrammingDependencies	31 01 FF 01	x	-	-	x	-
RequestDownload	34					
No sub function parameter	-	x	-	-	x	-
TransferData	36					
Block Sequence Counter	36 xx	x	-	-	x	-
RequestTransferExit	37					
No sub function parameter	-	x	-	-	x	-
TesterPresent	3E					
ZeroSubFunction	00	x	x	x	x	x
ControlDTCSetting	85					

DTCSettingType=on	85 01	x	x	-	-	x
DTCSettingType=off	85 02	x	x	-	-	x

Table 6-1Diagnostic services in boot software

6.3.1.1 Diagnostic Service with Sub Function Parameter

The following table shows the diagnostic services with sub function parameter that are supported in the application software. These services shall support the suppressPosRspMsgIndication-Bit. The ECU shall strictly keep to the status of the flag in the request message. It is in the responsibility of the tester to use the flag in connection with those services that do not require data from the ECU in the positive response.

Diagnostic Service Description	**SID (Hex Value)**
DiagnosticSessionControl	10
ECUReset	11
SecurityAccess	27
CommunicationControl	28
RoutineControl	31
TesterPresent	3E
ControlDTCSetting	85

Table 6-2Diagnostic services with sub function parameters

6.3.2 Diagnostic Services in Application Software

There are several diagnostic services that are necessary in the application software to support flash reprogramming. These services are shown in Table 6-3.

Diagnostic Service Description	**Hex vale**	**Phys.Req.**	**Funct.Req .**	**Supported in session**	
Subfunction Parameter				**$01**	**$03**
DiagnosticSessionControl	10				
defaultSession	10 01	X	X	X	X
programmingSession	10 02	X	-	-	X
extendedDiagnosticSession	10 03	X	X	X	X
ECUReset	11				
Power On Reset	11 01	X	X	X	X
ReadDataByIdentifier	22				
Data Identifier	22 xx yy	X	-	X	X
CommunicationControl	28				

disableRxAndTx	28 03 01	X	X	-	X
enableRxAndTx	28 00 01	X	X	-	X
ControlDTCSetting	85				
DTCSettingType=on	85 01	X	X	-	X
DTCSettingType=off	85 02	X	X	-	X

Table 6-3 Diagnostic services in application software

6.3.3 Diagnostic Services and Formats Supported in the Bootloader

This chapter specifies the diagnostic sub function parameters and message formats that are supported in the bootloader. This is a subset of the diagnostic communication services.

6.3.3.1 DiagnosticSessionControl $10

The service format is applied as specified in diagnostics specification but shall be restricted to the parameter vales in Table 6-4.

Service request message definition for $10.

Data Byte	Parameter Name	Hex Value
#0	DiagnosticSessionControl Request Service Id	10
#1	DiagnosticSessionType default Session programming Session extended Session	 01 02 03

Table 6-4DiagnosticSessionControl– Request Message

Positive response message format for $10.

Data Byte	Parameter Name	Hex Value
#0	Diagnostic Session Control Response Service Id	50
#1	DiagnosticSessionType default Session programming Session extended Session	 01 02 03
#2-5	SessionParameterRecord P2(high byte) P2(low byte) P2*(high byte) P2*(low byte)	 00-FF 00-FF 00-FF 00-FF

Table 6-5DiagnosticSessionContro-Positive response format

Negative response message format for $10.

Data Byte	Parameter Name	Hex Value

#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	10
#3	Negative Response Code	See below

Table 6-6DiagnosticSessionControl– Negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Subfunction not supported/Invalid format
13	Incorrect message length or invalid format
22	Conditions not correct or request sequence error
78	Request correctly received response Pending

6.3.3.2 ECUReset $11

The sub function parameter "Reset Mode" supported in the bootloader is "PowerOnReset"($01). Other values are not supported.

Service request message definition for $11.

Data Byte	Parameter Name	Hex Value
#0	ECU Reset Request Service Id	11
#1	ResetType power on reset	 01

Table 6-7ECUReset– Request Message

Positive response message format for $11.

Data Byte	Parameter Name	Hex Value
#0	ECU Reset Response Service Id	51
#1	Reset Type power on reset	 01

Table 6-8ECUReset– Positive response format

Negative response message format for $11.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	11
#2	Negative Response Code	See below

Table 6-9ECUReset– Negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Subfunction not supported Send if the sub-function parameter in the request message is not supported.
13	Incorrect message length or invalid format

	The length of the message is wrong.
22	Conditions not correct This code shall be returned if the criteria for the ECUReset request are not met.
78	Request correctly received response Pending

6.3.3.3 ReadDataByIdentifier $22

The service format is applied as specified in[1]. In the bootloader, only $F1 $83 –"ECUBootloaderSoftwareReferenceNumber".

6.3.3.3.1 ReadDataByIdentifier- "ECUBootloaderSoftwareReferenceNumber"

Service request message definition for data identifier $F1 $83.

Data Byte	Parameter Name	Hex Value
#0	ReadDataByIdentifier Request Service Id	22
#1-#2	DataIdentifier ECUBootloaderSoftwareReferenceNumber	F183

Table 6-11ReadDataByIdentifier-"ECUBootloaderSoftwareReferenceNumber"request format

Positive response format for data identifier $F1 $83.

Data Byte	Parameter Name	Hex Value
#0	ReadDataByIdentifierResponse Service Id	62
#1-#2	DataIdentifier ECUBootloaderSoftwareReferenceNumber	F183
#3	ECUName(Byte 0,ASCII-Coded)	41-5A
#4	ECUName(Byte 1,ASCII-Coded)	41-5A
#5	ECUName(Byte 2,ASCII-Coded)	41-5A
#6	VersionReleaseData YY(Byte 0,BCD-Coded)	00-99
#7	VersionReleaseDataYY(Byte 1,BCD-Coded)	00-99
#8	VersionReleaseDataMM(Byte 2,BCD-Coded)	00-12
#9	VersionReleaseDataDD(Byte 3,BCD-Coded)	00-31
#10	SoftwareVersionNumber(Byte 0,ASCII-Coded)	30-39,2E
#11	SoftwareVersionNumber(Byte 1,ASCII-Coded)	30-39,2E
#12	SoftwareVersionNumber(Byte 2,ASCII-Coded)	30-39,2E

Table 6-12ReadDataByIdentifier-"ECUBootloaderSoftwareReferenceNumber" positive response format

6.3.3.3.2 ReadDataByIdentifier- "Fingerprint"

Service request message definition for data identifier $F1 $84.

Data Byte	Parameter Name	Hex Value
#0	ReadDataByIdentifier Request Service Id	22

#1-#2	DataIdentifier Fingerprint	 F184

Table 6-13ReadDataByIdentifier-“Fingerprint” request format

Positive response format for data identifier $F1 $84.

Data Byte	Parameter Name	Hex Value
#0	ReadDataByIdentifierResponse Service Id	62
#1-#2	DataIdentifier Fingerprint	 F184
#3	ProgrammingData YY(Byte 0,BCD-Coded)	00-99
#4	ProgrammingData MM(Byte 1,BCD-Coded)	00-12
#5	ProgrammingData DD (Byte 2,BCD-Coded)	00-31
#6-#11	TesterSerialNumber (byte 0-5,HEX-Coded)	00-FF

Table 6-14ReadDataByIdentifier-“Fingerprint” positive response format

Negative response format for data identifier $22.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	22
#2	Negative Response Code	See below

Table 6-15ReadDataByIdentifier- negative response format

Negative Response Codes:

NRC(hex)	Error text
13	Incorrect message length or invalid format This response code shall be sent if the length of the request message is invalid.
22	Conditions not correct This response code shall be sent if the operating conditions of the server are not met to perform the required action.
31	Request out of range This code shall be sent if 1 none of the requested data identifier value are supported by the device. 2 the client exceeded the maximum number of data identifiers allowed to be requested at a time.
33	Security access denied This code shall be sent if at least one of the data identifiers is secured and the server is not in an unlocked state.
78	Request correctly received response Pending

6.3.3.4 Security Access $27

The security access service in the bootloader software uses security level code $11 for the request of a reprogramming seed and $12 to send the reprogramming key. Other security level codes are not supported in the bootloader. Additional requirements of security access are specified in Annex A. The following services shall only be accessible after a successful security access unlock procedure:$2E,$31–checkRoutine,$31–EraseMemory,$31–checkReprogrammingDependencies, $34, $36, $37.

Data Byte	Parameter Name	Hex Value
#0	Security Access Request Service Id	27
#1	Security Level Request Reprogramming Seed Send Reprogramming Key	 11 12
#2-#5	Security Access Data Record Access Mode = RequestSeed: no data record Access Mode = SendKey: Data record contains N key bytes	 00-FF

Table 6-16 Security Access request message definition

6.3.3.4.1 SecurityAccess–"RequestSeed"

Service request message definition for subfunction $11 –"requestSeed".

Data Byte	Parameter Name	Hex Value
#0	Security Access Request Service Id	27
#1	Security Level Request Reprogramming Seed	 11

Table 6-17 Security Access-"requestSeed"service request format

Positive response message format for subfunction $11 –"requestSeed".

Data Byte	Parameter Name	Hex Value
#0	Security Access Response Service Id	67
#1	Security Level Request Reprogramming Seed	 11
#2-5	Security Access Data Record SecuritySeed, 4 bytes	 00-FF

Table 6-18 Security Access-"requestSeed"response format

Negative response format for data identifier $27–"requestSeed".

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	27
#2	Negative Response Code	See below

Table 6-19Security Access- negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Sub-function not supported Send if the sub-function parameter in the request message is not supported.
13	Incorrect message length or invalid format

	The length of the message is wrong.
22	Conditions not correct The code shall be returned if the criteria for the request Security Aeccess are not met.
37	Required time delay not expired Send if the delay timer is active and a request is transmitted.

6.3.3.4.2 SecurityAccess–"SendKey"

Service request message definition for subfunction $12 –"sendKey".

Data Byte	Parameter Name	Hex Value
#0	Security Access Request Service Id	27
#1	Security Level Send Reprogramming Key	 12
#2-5	SecurityKey,4 bytes	00-FF

Table 6-20 Security Access-"SendKey"request format

Positive response message format for subfunction $12 –"SendKey".

Data Byte	Parameter Name	Hex Value
#0	Security Access Response Service Id	67
#1	Security Level Send Reprogramming Key	 12

Table 6-21 Security Access-"SendKey"response format

Negative response format for data identifier $27–"SendKey".

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	27
#2	Negative Response Code	See below

Table 6-22Security Access- negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Sub-function not supported Send if the sub-function parameter in the request message is not supported.
13	Incorrect message length or invalid format The length of the message is wrong.
22	Conditions not correct The code shall be returned if the criteria for the request Security Aeccess are not met.
24	Request sequence error Send if the 'sendkey' sub-function is received without first receiving a 'requestseed' request message.
35	Invalid key Send if an expected 'sendkey' sub-function value is received and the value of the key does

	not match the server's internally stored/calculated key.
36	Exceeded number of attempts Send if the delay timer is active due to exceeding the maximum number of allowed false access attempts.
37	Required time delay not expired Send if the delay timer is active and a request is transmitted.

6.3.3.5 CommunicationControl$28

This service format is applied as specified in [1] but shall be restricted to the parameter vales in Table 6-17.

Data Byte	Parameter Name	Hex Value
#0	CommunicationControlRequest Service Id	28
#1	ControlType enableRxAndTx disableRxAndTx	 00 03
#2	CommunicationType	01

Table 6-23CommunicationControl - Request format

Positive response message format for $28.

Data Byte	Parameter Name	Hex Value
#0	CommunicationControl Response Service Id	68
#1	ControlType enableRxAndTx disableRxAndTx	 00 03

Table 6-24CommunicationControl– Positive response format

Negative response format for data identifier $28.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	28
#2	Negative Response Code	See below

Table 6-25CommunicationControl- negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Sub-function not supported Send if the sub-function parameter in the request message is not supported.
13	Incorrect message length or invalid format The length of the message is wrong.
22	Conditions not correct The code shall be returned if the criteria for the request Security Aeccess are not met.

31	Request out of range The server shall use this response code,if it detects an error in the communication type parameter.
78	Request correctly received response Pending

6.3.3.6 WriteDataByIdentifier $2E

The WriteDataByIdentifier service in the bootloader supports the data identifier $F1 $84 –“Fingerprint”, and has got the following format.

Data Byte	Parameter Name	Hex Value
#0	WriteDataByIdentifier Request Service Id	2E
#1	Fingerprint DID (MSB)	F1
#2	Fingerprint DID (LSB)	84
#3	ProgrammingData YY(Byte 0,BCD-Coded)	00-99
#4	ProgrammingData MM(Byte 1,BCD-Coded)	00-12
#5	ProgrammingData DD (Byte 2,BCD-Coded)	00-31
#6-#11	TesterSerialNumber(byte 0-5,HEX-Coded)	00-FF

Table 6-26WriteDataByIdentifier–“Fingerprint”

Positive response message format for $2E.

Data Byte	Parameter Name	Hex Value
#0	WriteDataByIdentifierResponse Service Id	6E
#1-#2	DataIdentifier Fingerprint	F184

Table 6-27WriteDataByIdentifier– Positive response format

Negative response format for data identifier $2E.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	2E
#2	Negative Response Code	See below

Table 6-28WriteDataByIdentifier- negative response format

Negative Response Codes:

NRC(hex)	Error text
13	Incorrect message length or invalid format The length of the message is wrong.
22	Conditions not correct This response code shall be sent if the operating conditions of the server are not met to perform the required action.
31	Request out of range

	This response code shall be sent if 1 the date identifier in the request message is not supported in the server or the data identifier is supported for read only purpose (via ReadByIdentifier service). 2 Any data transmitted in the request message after the data identifier is invalid (if applicable to the node).
33	Security access denied This code shall be sent if at least one of the data identifiers is secured and the server is not in an unlocked state.
72	General programming failure This return code shall be sent if the server detects an error when writing to a memory location.

6.3.3.7 RoutineControl $31

The RoutineControl service in the bootloader is supported for the routine identifiers $02 $02 –"checkRoutine", $02 $03 –"checkProgrammingPreconditions", $FF $00 –"eraseMemory" and $FF $01 –"checkProgrammingDependencies".

The bootloader software shall only support routineControlType $01 –"startRoutine".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Request Service Id	31
#1	RoutineControl Type StartRoutine	 01
#2-#3	RoutineIdentifier CheckRoutine CheckProgrammingPreconditions EraseMemory CheckProgrammingDependencies	 0202 0203 FF00 FF01
#4-n	RoutineControlOptionRecord	00-FF

Table 6-29RoutineControl service request format

6.3.3.7.1 RoutineControl–"checkRoutine"

Service request message definition for routine identifier $02 $02 –"checkRoutine".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Request Service Id	31
#1	RoutineControl Type StartRoutine	 01
#2-#3	RoutineIdentifier CheckRoutine	 0202

#4-#7	CRC32 value,4 bytes	00-FF

Table 6-30RoutineControl-"checkRoutine"service request format

Positive response format for routine identifier \$02 \$02 –"checkRoutine".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Response Service Id	71
#1	RoutineControl Type StartRoutine	 01
#2-#3	RoutineIdentifier CheckRoutine	 0202
#4	Routine Status Record Correct Result Incorrect Result	 00 01

Table 6-31RoutineControl–"CheckRoutine" response format

6.3.3.7.2 RoutineControl–"checkProgrammingPreconditions"

Service request message definition for routine identifier \$02 \$03 –"checkProgrammingPreconditions"

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Request Service Id	31
#1	RoutineControl Type StartRoutine	 01
#2 - #3	RoutineIdentifier CheckProgrammingPreconditions	 0203

Table 6-32RoutineControl – "checkProgrammingPreconditions" service request format

Positive response format for routine identifier \$02 \$03 –"checkProgrammingPreconditions".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Request Service Id	71
#1	RoutineControl Type StartRoutine	 01
#2 - #3	RoutineIdentifier CheckProgrammingPreconditions	 0203

Table 6-33RoutineControl – "checkProgrammingPreconditions" response format

6.3.3.7.3 RoutineControl–"eraseMemory"

Service request message definition for routine identifier \$FF \$00 – "eraseMemory".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Request Service Id	31
#1	RoutineControl Type StartRoutine	 01

#2-#3	RoutineIdentifier eraseMemory	 FF00
#4	addressAndLengthFormatIdentifier lengthFormat: bit 7 – 4: number of bytes of the memorySize parameter addressFormat: bit 3- 0: number of bytes of the memoryAddressparameter Recommended fix value	 0x-4x X0-x4 44
#5 – n1	memoryAddress 1 – 4 bytes erase address	00 - FF
n2 – n3	memorySize 1 – 4 bytes erase size	00 - FF

Table 6-34RoutineControl – "eraseMemory" service request format

Positive response format for routine identifier $FF $00 – "eraseMemory".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Request Service Id	71
#1	RoutineControl Type StartRoutine	 01
#2-#3	RoutineIdentifier eraseMemory	 FF00
#4	routineStatusRecord correctResult incorrectResult	 00 01

Table 6-35RoutineControl – "eraseMemory" response format

6.3.3.7.4 RoutineControl–"checkProgrammingDependencies"

Service request message definition for routine identifier $FF $01 –"checkProgrammingDependencies".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Request Service Id	31
#1	RoutineControl Type StartRoutine	 01
#2-#3	RoutineIdentifier CheckProgrammingDependencies	 FF01

Table6-36RoutineControl-"CheckProgrammingDependencies"service request format

Positive response format for routine identifier $FF$01 –"CheckProgrammingDependencies".

Data Byte	Parameter Name	Hex Value
#0	RoutineControl Response Service Id	71
#1	RoutineControl Type StartRoutine	 01
#2-#3	RoutineIdentifier	

	CheckProgrammingDependencies	FF01
#4	Routine Status Record Correct Result Incorrect Result	 00 01

Table6-37RoutineControl-“CheckProgrammingDependencies”serviceresposne format

Negative response format for data identifier $31.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	31
#2	Negative Response Code	See below

Table 6-38RoutineControl- negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Sub-function not supported This code is returned if the requested sub-function is not supported.
13	Incorrect message length or invalid format The length of the message is wrong.
22	Conditions not correct The code shall be returned if the criteria for the request RoutineControl are not met.
24	Request sequence error This code shall be returned if the ‘stopRoutine’ or ‘requestRoutineResults’ sub-function is received without first receiving a ‘startRoutine’ for the requested routine identifier.
31	Request out of range This code shall be returned if 1 the server does not support the requested routine identifier. 2 the user optional routineControlOptionRecord contains invalid data for the requested routine identifier.
33	Security access denied This code shall be sent if this code shall be returned if a client sends a request with a valid secure routine identifier and the server’s security feature is currently active.
70	Upload Download Not Accepted This response code indicates that an attempt to download to a server’s memory cannot be Accomplished due to some fault conditions.
72	General programming failure This return code shall be sent if the server detects an error when performing a routine,which accesses server internal memory.An example is when the routine erases or programs a certain memory location in the permanent device (e.g. Flash Memory) and the access to that memory location fails.
78	Request correctly received response Pending

6.3.3.8 RequestDownload $34

The service format is applied as specified in followings.

Service request message definition for $34.

Data Byte	Parameter Name	Hex Value
#0	RequestDownloadRequest Service Id	34
#1	dataFormatIdentifier	00
#2	addressAndLengthFormatIdentifier	44
#3-#6	memoryAddress(byte 0-3)	00-FF
#7-#10	memorySize(byte 0-3)	00-FF

Table 6-39RequestDownload– Request Message

Positive response message format for $34.

Data Byte	Parameter Name	Hex Value
#0	RequestDownload Response Service Id	74
#1	lengthFormatIdentifier	20
#2-#3	maxNumberofBlockSize(byte 0-1)	00-FF

Table 6-40RequestDownload– Positive response format

Negative response format for data identifier $34.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	34
#2	Negative Response Code	See below

Table 6-41RequestDownload - negative response format

Negative Response Codes:

NRC(hex)	Error text
13	Incorrect message length or invalid format The length of the message is wrong.
22	Conditions not correct The return code shall be sent if a server receives a request for this service while in the process of receiving a download of a software or calibration module. This could occur if there is a data size mismatch between the server and the client during the download of a module.
31	Request out of range This return code shall be sent if 1 the specified dataFormatIdentifier is not valid. 2 the specified address and LengthFormatIdentifier is not valid. 3 the specified memory Address/memory Size is not valid.
33	Security access denied This return code shall be sent if the server is secure (for server's that support the security access service) when a request for this service has been received.

70	Upload download not accepted This response code indicates that an attempt to download to a server’s memory cannot be accomplished due to some fault conditions.
78	Request correctly received response Pending

6.3.3.9 TransferData $36

The service format is applied as specified in followings. The block sequence counter shall be treated as specified in [2].

Service request message definition for $36.

Data Byte	Parameter Name	Hex Value
#0	TransferDataRequest Service Id	36
#1	blockSequenceCounter	00-FF
#2-#(2+n-1)	transferRequestParameterRecord Parameter 0 . . . Parameter(n-1)	 00-FF . . . 00-FF

Table 6-42 TransferData– Request Message

Positive response message format for $36.

Data Byte	Parameter Name	Hex Value
#0	TransferDataResponse Service Id	76
#1	blockSequenceCounter	00-FF

Table 6-43 TransferData– Positive response format

Negative response format for data identifier $36.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	36
#2	Negative Response Code	See below

Table 6-44TransferData - negative response format

Negative Response Codes:

NRC(hex)	Error text
13	Incorrect message length or invalid format The length of the message is wrong.
24	Request sequence error The server shall use this response code 1 if the request download or request upload service is not active when a request for this service is received.

	2 if the request download or request upload service is active, but the server has already received al data as determined by the memory size parameter in the active request download or request upload service. NOTE The repetition of a Transfer Data request message with a block sequence counter equal to the one included in the previous transfer data request message shall be accepted by the server.
31	Request out of range The return code shall be sent if the transfer request parameter record contains additional control parameters and this control information is invalid.
33	Security access denied This code shall be sent if this code shall be returned if a client sends a request with a valid secure routine identifier and the server's security feature is currently active.
71	Transfer data suspended This return code shall be sent if 1 the response code indicates that a data transfer operation was halted due to some fault. 2 the download module length does not meet the requirements of the memory size parameter sent in the request message of the request download service.
72	General programming failure This return code shall be sent if the server detects an error when erasing or performing a memory location in the permanent memory device during the download of data.
73	Wrong block sequence counter This return code shall be sent if the server detects an error in the sequence of the block-sequence counter. NOTE The repetition of a transfer data request message with a block sequence counter equal to the one included in the previous transfer data request message shall be accepted by the server.
92-93	Voltage too high/voltage too low This return code shall be sent as applicable if the voltage measured at the primary power pin of the server is out of the acceptable range for downloading data into the server's permanent memory.
78	Request correctly received response Pending

6.3.3.10 RequestTransferExit $37

The service format is applied as specified in followings.

Service request message definition for $37.

Data Byte	Parameter Name	Hex Value
#0	RequestTransferExitRequest Service Id	37

Table 6-45RequestTransferExit– Request Message

Positive response message format for $37.

Data Byte	Parameter Name	Hex Value
#0	RequestTransferExitResponse Service Id	77

Table 6-46RequestTransferExit– Positive response format

Negative response format for data identifier $37.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	37
#2	Negative Response Code	See below

Table 6-47RequestTransferExit - negative response format

Negative Response Codes:

NRC(hex)	Error text
13	Incorrect message length or invalid format The length of the message is wrong.
24	Request sequence error This return code shall be sent if: 1 the programming process is not completed when a request for this service is received. 2 the request download or request upload service is not active.

6.3.3.11 TesterPresent $3E

The service format is applied as specified in followings.

Service request message definition for $3E.

Data Byte	Parameter Name	Hex Value
#0	TesterPresentRequest Service Id	3E
#1	zeroSubFunction	00

Table 6-48TesterPresent– Request Message

Positive response message format for $3E.

Data Byte	Parameter Name	Hex Value
#0	TesterPresentResponse Service Id	7E
#1	zeroSubFunction	00

Table 6-49TesterPresent– Positive response format

Negative response format for data identifier $3E.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	3E
#2	Negative Response Code	See below

Table 6-50TesterPresent- negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Sub-function not supported Send if the sub-function parameter in the request message is not supported.
13	Incorrect message length or invalid format The length of the message is wrong.

6.3.3.12 ControlDTCSetting $85

The service format is applied as specified in followings.

Service request message definition for $85.

Data Byte	Parameter Name	Hex Value
#0	ControlDTCSettingRequest Service Id	85
#1	DTCSettingType on off	 01 02

Table 6-51ControlDTCSetting– Request Message

Positive response message format for $85.

Data Byte	Parameter Name	Hex Value
#0	ControlDTCSettingResponse Service Id	C5
#1	DTCSettingType on off	 01 02

Table 6-52ControlDTCSetting– Positive response format

Negative response format for data identifier $85.

Data Byte	Parameter Name	Hex Value
#0	Negative Response Service Identifier	7F
#1	Original Request Service Identifier	85
#2	Negative Response Code	See below

Table 6-53ControlDTCSetting - negative response format

Negative Response Codes:

NRC(hex)	Error text
12	Sub-function not supported Send if the sub-function parameter in the request message is not supported.
13	Incorrect message length or invalid format The length of the message is wrong.
22	Conditions not correct

	Used when the server is in a critical normal mode activity and therefore cannot perform the requested DTC control functionality.
31	Request out of range The server shall use this response code, if it detects an error in the DTC setting control option record.

7 Reprogramming Process Related Data Requirements

The flash process requires storing a set of non-volatile data. This chapter is a proposal for how to handle this information.

The non-volatile reprogramming information block consists of two parts - the general bootloader status information and the meta data table. The general information block stores flags according to reprogramming requests, validity status of logical blocks, etc. The meta data table provides information for each logical block like reprogramming counter and CRC.

7.1 General Non-Volatile Bootloader Status Information

The system shall store the following fields in the non-volatile memory(EEPROM recommend). Each field in Table7-1 is mandatory, but the in-memory order is an example.

Offset/Bytes	Size/Bytes	Field	Coding/Hex	Description
+0	1	External reprogramming request flag	00/FF B5	No external reprogramming request present The external reprogramming request flag indicates that the ECU application started the bootloader for reprogramming.
			01-B4 B6-FE	Reserved Reserved
+1	N^2	Validity Flags	b7-b0	Inverted validity bits of all logical blocks

			bx=1
			bx=0
				Logical block valid
				Logical block invalid

+2+n	1	Security Access Delay Flag	0 1	Security access delay not active Security access delay active

Table 7-1 General non-volatile bootloader status information

The handling of validity information is shown in Figure 7-2 in more detail. Validity information is used by the bootloader to determine if the application can be started after PowerOn/Reset. Depending on the number of logical blocks, there are n bytes reserved to store this information. Each logical block is represented by one bit as shown in Figure 7-2.

The bits b7 – b0 represent the validity of the corresponding logical block. If more than eight logical blocks are configured, additional bytes according to this schema are used.

7.2 Meta Data Table

For each logical block, the system shall store the information specified by Table 7-2.

Offset/Bytes	Size/Bytes	Field	Coding/Hex	Description	non-volatile memory type
0	9	Fingerprint	00-FF	Flash reprogramming fingerprint	EEPROM
9	2	Programming Counter	0000-FFFF	Counter for every reprogramming event of each logic block	EEPROM
11	4	CRC	32bit,Binary	CRC32 of block data	Flash of each logic block
15	4	CRCstart	32bit,Binary	Start address for CRC calculation	Flash of each logic block
19	4	CRClength	32bit,Binary	Data length for CRC calculation	Flash of each logic block

Table 7-2 Non-volatile information for each logical block

Offset/Bytes	Size/Bytes	Field	Coding/Hex	Description
0	3	Programming Data High Byte:year Mid Byte:Month Low Byte:Day	 BCD BCD BCD	Programming Data
3	6	Tester Number	00-FF	Tester serial number

Table 7-3 Fingerprint format

The fingerprint format is specified in Table 7-3.

Annex A

(implementation Rules of SecurityAccess ($27))

After PowerOn/Reset the ECU is in locked state. The security access failure counter is set to 0.The ECU shall wait 10 seconds before accepting the first RequestSeed message after EcuReset/PowerOn. The implementation of this behaviour shall be agreed by BAIC EV. Any SecurityAccess request during this time will be rejected with the negative response code“Required time delay not expired” (37).If the tester requests a seed, it has to send the corresponding key to the ECU. This sequence is mandatory. If the tester sends a consecutive “Request Seed”, the request is accepted and the same seed is returned, but the security access failure counter is incremented. If the tester sends an invalid key, the request is rejected with negative response code “InvalidKey”, the sequence shall be reset (any current seed becomes invalid) and the security access failure counter is incremented. When the security counter reaches the value of 3 (i.e. 3 failed tries), the ECU shall wait 10s before accepting another “Request Seed” message. Any such request during this time will be rejected with the negative response code “Required time delay not expired” (37). When the 10s wait time is elapsed the security access failure counter is decremented by one and another try is allowed. When during this try the security access failure counter is incremented again (due to an invalid key), the ECU shall wait again 10s before accepting another “Request Seed” message. When this try is valid, the security access failure counter is not changed.

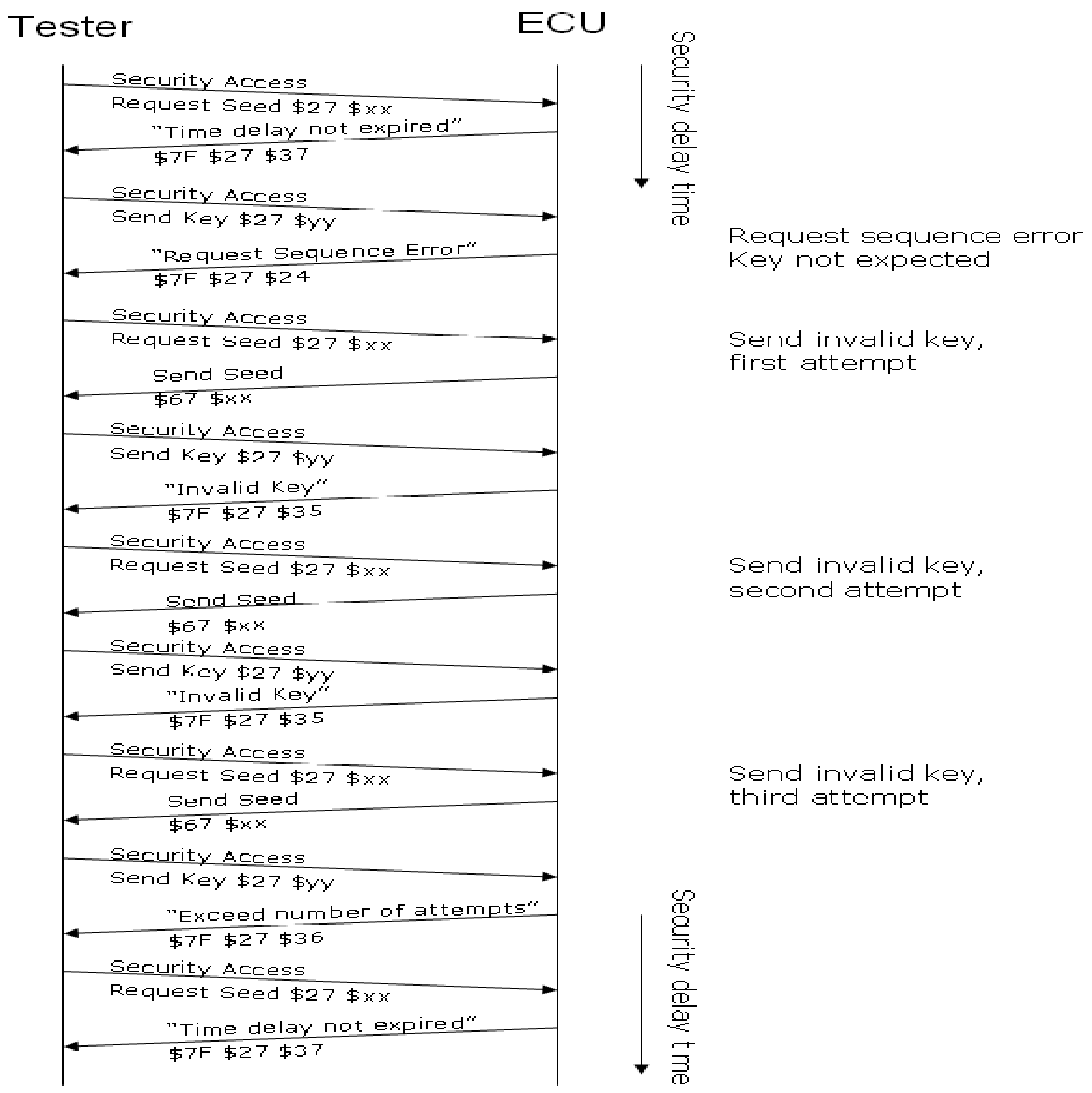


Figure -1: SecurityAccess NRC requirements

Secured data identifiers require an increased security level. Therefore, they are accessible only within a non-default diagnostic session. Secured memory areas require a SecurityAccess service and therefore a non-default diagnostic session. Secured routines require a SecurityAccess service and therefore a non-default diagnostic session. A routine that requires to be stopped actively by the client also requires a non default session.

Annex B

(programming message flow example)

A typical programming flow example is showed below which include pre-programming step, Server- programming step and post-programming step. The condition is that ECU physical request ID is 0x7E0, ECU physical response ID is 0x7E8, Functional request ID is 0x7DF.

● pre-programming step(only supported by app)

Some of diagnostic commands of pre-programming step are only supported by APP. If APP is non-existent in ECU, these commands will be received by bootloader, but bootloader do not support some of these commands. So responds will differ between APP and bootloader.

Table B.1 enter extern session (APP or BOOT)

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7DF	Client	02	10	03	AA	AA	AA	AA	AA
7E8	Server	06	50	03	00	32	00	C8	AA

This diagnostic command is also supported by bootloader. So the responds are almost the same.

Table B.2 check the programming condition (APP)

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	04	31	01	02	03	AA	AA	AA
7E8	Server	04	71	01	02	03	AA	AA	AA

This diagnostic command is not supported by bootloader. So below table is the respond of BOOT.

Table B.2 check the programming condition (BOOT)

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	04	31	01	02	03	AA	AA	AA
7E8	Server	03	7F	31	7F	AA	AA	AA	AA

Table B.3 disable detection and storage of DTCs (APP)

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7DF	Client	02	85	02	AA	AA	AA	AA	AA
7E8	Server	02	C5	02	AA	AA	AA	AA	AA

This diagnostic command is not supported by bootloader. But the command is functional request, so non-respond will be given by BOOT. More detail information is specified in [2].

Table B.4 disable transmission of normal, non-diagnostic messages(APP)

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7DF	Client	03	28	03	01	AA	AA	AA	AA
7E8	Server	02	68	03	AA	AA	AA	AA	AA

This diagnostic command is not supported by bootloader yet. But the command is functional request, so non-respond will be given by BOOT. More detail information is specified in [2].

Table B.5 read the DID F183 (BOOT version)(APP or BOOT)

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	03	22	F1	83	AA	AA	AA	AA
7E8	Server	10	0D	62	F1	83	56	42	55
7E0	Client	30	08	14	AA	AA	AA	AA	AA
7E8	Server	21	20	14	07	17	34	30	30

- ## server-programming step

Table B.6 enter the programming session

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	02	10	02	AA	AA	AA	AA	AA
7E8	Server	03	7F	10	78	AA	AA	AA	AA
7E8	Server	06	50	02	00	32	00	C8	AA

Table B.7 unlock ECU

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	02	27	11	AA	AA	AA	AA	AA
7E8	Server	06	67	11	**	**	**	**	AA
7E0	Client	06	27	12	**	**	**	**	AA
7E8	Server	02	67	12	AA	AA	AA	AA	AA

Table B.8 write the fingerprint

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	10	0C	2E	F1	84	00	00	00
7E8	Server	30	08	01	AA	AA	AA	AA	AA
7E0	Client	21	00	00	00	00	00	00	AA
7E8	Server	03	6E	F1	84	AA	AA	AA	AA

➢ **start to download the flash driver**

Table B.9 request download data

CAN ID	Client/ Server	Bytes							
		#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	10	0B	34	00	44	D4	00	00
7E8	Server	30	08	01	AA	AA	AA	AA	AA
7E0	Client	21	00	00	00	00	CE	AA	AA
7E8	Server	04	74	20	04	02	AA	AA	AA

Table B.10 transfer data

CAN ID	Client/ Server	Bytes							
		#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	10	D0	36	01	B7	04	10	F0
7E8	Server	30	08	01	AA	AA	AA	AA	AA
7E0	Client	21	00	00	00	00	00	00	00
7E0	Server	22	00	00	00	00	00	00	00
:									
7E0	Server	2D	00	00	00	00	00	00	AA
7E8	Server	03	7F	36	78	AA	AA	AA	AA
7E8	Server	02	76	01	AA	AA	AA	AA	AA

Table B.11 Request Transfer Exit

CAN ID	Client/ Server	Bytes							
		#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	01	37	AA	AA	AA	AA	AA	AA
7E8	Server	01	77	AA	AA	AA	AA	AA	AA

Table B.12 Start check Routine integrity of the downloaded data

CAN ID	Client/ Server	Bytes							
		#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	10	08	31	01	02	02	02	AB
7E8	Server	30	08	01	AA	AA	AA	AA	AA
7E0	Client	21	E3	6F	AA	AA	AA	AA	AA
7E8	Server	03	7F	31	78	AA	AA	AA	AA
7E8	Server	05	71	01	02	02	00	AA	AA

➢ **download the flash driver is over**

Table B.13 Start erase logic Memory areas of APP

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	10	08	31	01	FF	00	44	A0
7E8	Server	30	08	01	00	00	00	00	00
7E0	Client	21	02	00	00	00	04	70	00
7E8	Server	03	7F	31	78	AA	AA	AA	AA
7E8	Server	05	71	01	FF	00	00	AA	AA

Table B.14 request download data

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	10	0B	34	00	44	A0	02	00
7E8	Server	30	08	01	AA	AA	AA	AA	AA
7E0	Client	21	00	00	00	00	74	00	00

Table B.15 transfer data

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	10	D0	36	01	B7	04	10	F0
7E8	Server	30	08	01	AA	AA	AA	AA	AA
7E0	Client	21	AA	AA	AA	AA	AA	AA	AA
7E0	Server	22	AA	AA	AA	AA	AA	AA	AA
:									
7E0	Server	2D	AA	AA	AA	AA	AA	AA	AA
7E8	Server	03	7F	36	78	AA	AA	AA	AA
7E8	Server	02	76	01	AA	AA	AA	AA	AA

Table B.16 Request Transfer Exit

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	01	37	AA	AA	AA	AA	AA	AA
7E8	Server	01	77	AA	AA	AA	AA	AA	AA

Table B.17 Start check Routine integrity of the downloaded data

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8

7E0	Client	10	08	31	01	02	02	02	00
7E8	Server	30	08	01	AA	AA	AA	AA	AA
7E0	Client	21	AA	AA	AA	AA	AA	AA	AA
7E8	Server	03	7F	31	78	AA	AA	AA	AA
7E8	Server	05	71	01	02	02	00	AA	AA

Table B.18 check Programming Dependencies

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	04	31	01	FF	01	AA	AA	AA
7E8	Server	03	7F	31	78	AA	AA	AA	AA
7E8	Server	05	71	01	FF	01	00	AA	AA

- **post-programming step**

Table B.19 ECU Reset

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7E0	Client	02	11	01	AA	AA	AA	AA	AA
7E8	Server	02	51	01	AA	AA	AA	AA	AA

After ECU resetting, program will jump from FBL to APP. So the below diagnose commands should be supported by APP.

Table B.20 enter extern session

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7DF	Client	02	10	03	AA	AA	AA	AA	AA
7E8	Server	06	50	03	00	32	00	C8	AA

Table B.21 enable transmission of normal, non-diagnostic messages

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7DF	Client	03	28	00	01	AA	AA	AA	AA
7E8	Server	02	68	00	AA	AA	AA	AA	AA

Table B.22 enable detection and storage of DTCs

CAN ID	Client/	Bytes							
	Server	#1	#2	#3	#4	#5	#6	#7	#8
7DF	Client	02	85	01	AA	AA	AA	AA	AA
7E8	Server	02	C5	01	AA	AA	AA	AA	AA

Table B.19 enter the default session

CAN ID	Client/ Server	Bytes							
		#1	#2	#3	#4	#5	#6	#7	#8
7DF	Client	02	10	01	AA	AA	AA	AA	AA
7E8	Server	02	50	01	AA	AA	AA	AA	AA

Annex C

（ detail pre-condition for programming ）

各 ECU 在遵循该规定的前提下，为了更好的规范整车级刷写规范，避免 ECU 刷程序时其他各 ECU 在刷写过程中报故障这一现象，采用 STATE 机制对所有 ECU 进行有效的管控。

以下为各节点的规范要求:

Each ECU must be in compliance with the specification of the premise, in order to regulate the vehicle level programming specification, and to avoid making other ECU generate the DTC when reprogramming an ECU, We use STATE effective mechanism to manage and control each ECU.

The following is the specification requirements of each node:

1、各 ECU 的 Bootloader 上位机要求：
each of ECU Bootloader tester requirements:

1) 严格遵循 Annex B 描述规范进行报文的发送；
Strictly follow the specifications described in Annex B to send and respond CAN frame.

2) 各 ECU 刷写设备必须支持 PCAN，配套上位机软件不做要求；
Each ECU should support PCAN device and PCAN-Explorer5 is not necessary;

2、各 ECU 软件要求：
each ECU software requirements:

1) 各 ECU 要求具备停止诊断功能、停止报文发送要求；
Each ECU should support the diagnostic service 28(CommunicationControl service) and 85(ControlDTCSetting service) ;

2) 各 ECU 收到 state255 后，应停止诊断、停止报文发送；

If received the state255, the ECU should disable the diagnosis function and stop sending CAN frame;

3) 各 ECU 收到 state255 后，应保存当前环境且 state 不可逆转，此状态持续到下电为止。

If received the state255,the ECU should be save the current environment and the state is irreversible, and should keep state255 until the next power on.

3、VCU 要求：

VCU requirements:

1) VCU 需要具备各相关网段内广播 state255 的功能；

VCU should have the function to broadcast state255 to the every relevant network;

2) VCU 需要具备是否符合刷写状态的判断；

VCU is required to determine whether the current state is approgriate for programming .

3) VCU 需具备 state255 到 state49 的跳转。

VCU can jump from state255 to state49.

4、总线要求：

CAN Bus requirements:

1) 整车相关 ECU 必须与 VCU 在总线上有通路；

Vehicle-related ECU must have access with VCU on the bus;

2) VCU 需要给所有整车相关 ECU 所在网段进行 STATE 的广播，广播周期保持 10ms。

VCU need broadcast STATE to the vehicle-related ECUs in the relevant networks, the broadcast cycle maintain 10ms.

5、具体步骤及规范：

Specific steps and specifications:

1) 当上位机7DF依次接收的内容为:02 10 03；02 85 02；03 28 03 01，VCU认为有ECU需要bootloader刷写，VCU判断整车状态。如果整车允许刷程序，VCU进入state255状态，同时广播state255到各网络；

When the tester for the first time send DiagnosticSessionControl $10 $03 , ControlDTCSetting $85 $02, and CommunicationControl $28 $03 $01 via function request, VCU should determine that an ECU needs to program APP via bootloader at this time and judge the current vehicle state. If the current state of vehicle allows program the APP, VCU need enter state255 state and broadcast state255 to every network;

2) 各 ECU 收到 255 广播帧后，应停止发送业务报文，停止故障诊断，此状态持续到下电为止；

After the receipt of 255state broadcast frames, each ECU must stop sending business messages frame, stop detect diagnostic; VCU need to continue broadcasting the state255;

3) 当 ECU-A 需要刷写程序，收到报文 7**后，ECU-A 判断 VCU 是否广播 255，如果没有，ECU-A 需等待 10S，若 10S 内没有 state255，表示整车不适合刷写，上位机直接超时跳出;

   When the ECU-A need to program, received function request like 7**, ECU-A need to determine whether the sate255 frame of VCU has been received ,if not, ECU-A need to wait 10 seconds . if ECU-A do not received the state255 within 10S, ECU-A will not give any responds to the tester tool and the test tool will stop programming.because the current condition of vehicle is not suitable for programming;

4) ECU-A 收到 state255 时，ECU-A 保存环境，重置 ECU 进入 bootloader 模式进行程序的刷写，此时 VCU 持续广播 state255；

   When the ECU-A receives state255, ECU-A must save environment, and reset the ECU into the bootloader mode to programming, then the VCU continued broadcasting state255;

5) ECU-A 刷写完毕重启进入正常模式时，接收 VCU 广播的 state255，并进入 255 状态，此时上位机显示刷写完成；此时需重新机械上下电生效；

   When ECU-A programming finished and rebooting into normal mode(APP), ECU-A receives the VCU broadcasted state255 and enters into the 255 state, then the tester display programming completely; At this point ECU-A need to become effective of Power On/Off again;

6) 当 VCU 自己刷写程序时，VCU 接收到 7C*时，需持续广播 255 状态 5 次，其他 ECU 都应确保能收到，并各自做 state255 的处理；

   When VCU programmed itself, VCU received 7C *, need to continue broadcasting 255 state five times, Other ECU should ensure the frame is received, and every ECU should process state 255;

7) VCU 刷写完毕重置后再次进入 state255，其他节点一致都处于 state255。

   VCU will enter into state 255 again after programmed completely and Consistent with all other nodes should enter into state255.

8) 针对任何节点刷写失败的情况，控制器应支持在不下电的情况下重复刷写；

   If any ECU fails to upgrade, the ECU should repeat upgrade in case of Vehicle power on.

9) 若任何节点在手动刷写失败时，必须下电上电后再刷写；对于 T-BOX 刷写，不做限制；

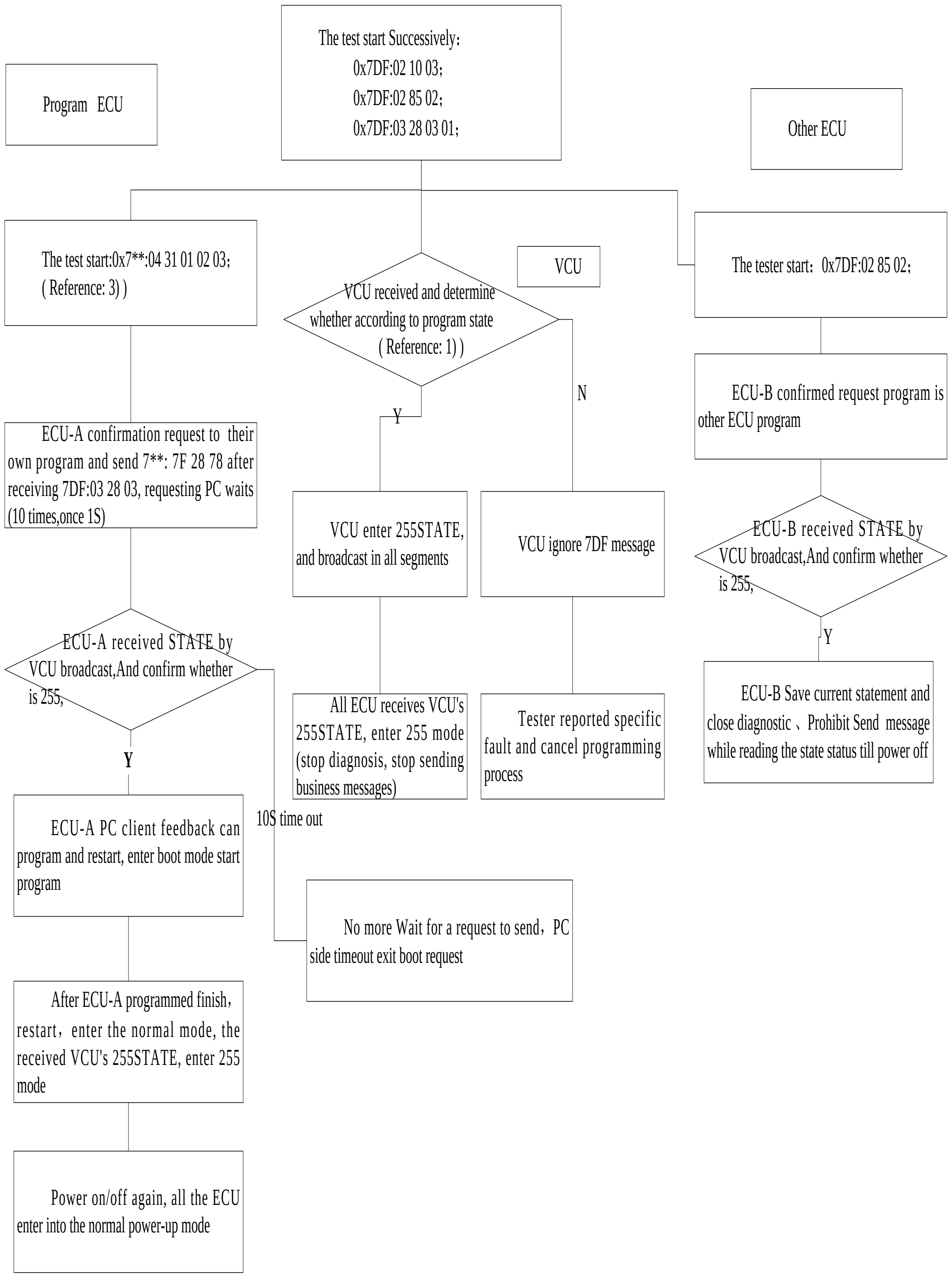
ALL ECU must restart by Vehicle power off/on when any manual upgrade failure, except T-BOX.

具体流程描述如下：

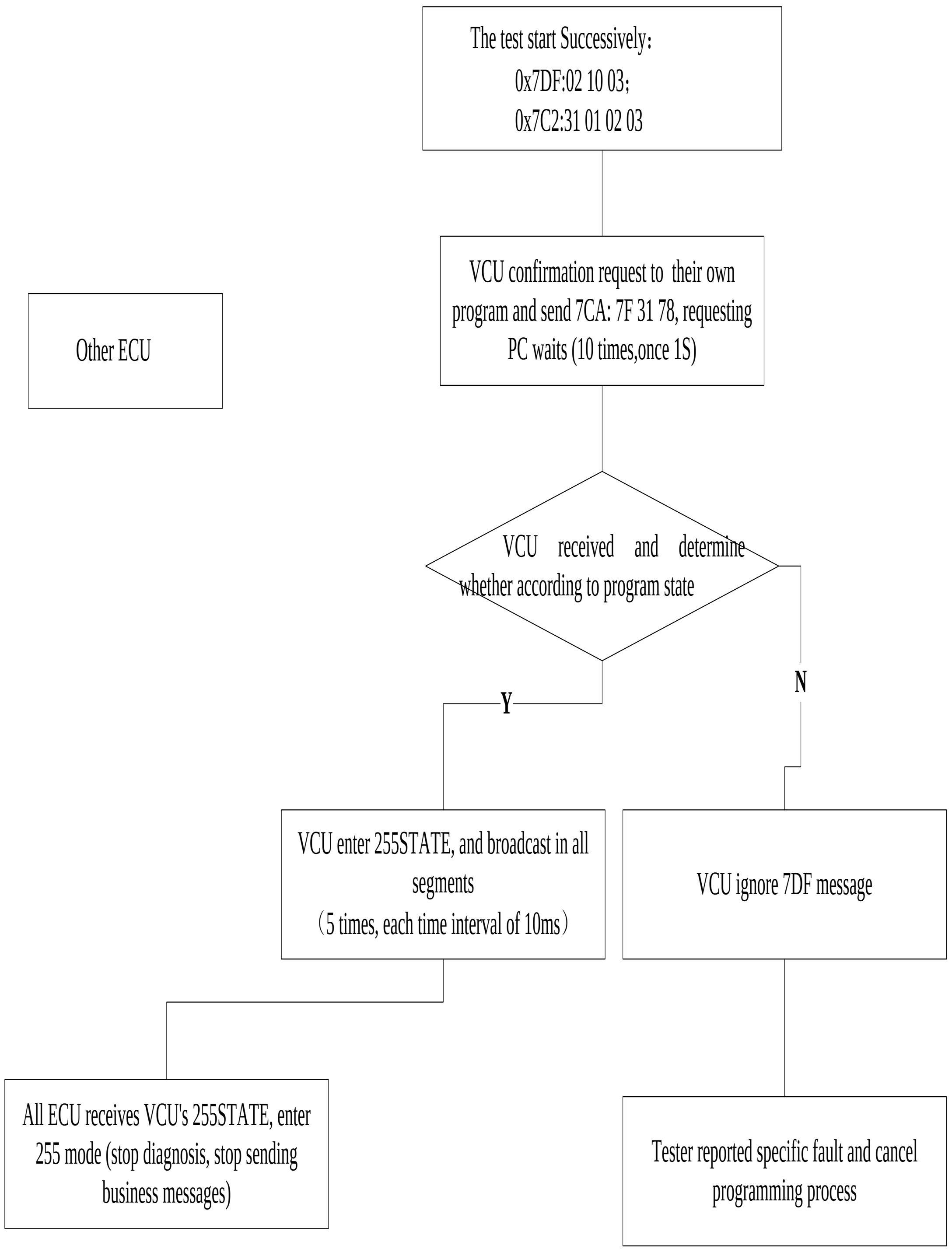
Specific processes are described as follows:

非 VCU 控制程序刷写流程

Non VCU control program process


Program ECU
The test start Successively:
0x7DF:02 10 03;
0x7DF:02 85 02;
0x7DF:03 28 03 01;
Other ECU
The test start:0x7**:04 31 01 02 03;
(Reference: 3))
VCU
VCU received and determine whether according to program state (Reference: 1))
The tester start：0x7DF:02 85 02;
N
Y
ECU-B confirmed request program is other ECU program
ECU-A confirmation request to their own program and send 7**: 7F 28 78 after receiving 7DF:03 28 03, requesting PC waits (10 times,once 1S)
VCU enter 255STATE, and broadcast in all segments
VCU ignore 7DF message
ECU-B received STATE by VCU broadcast,And confirm whether is 255,
Y
ECU-A received STATE by VCU broadcast,And confirm whether is 255,
All ECU receives VCU's 255STATE, enter 255 mode (stop diagnosis, stop sending business messages)
Tester reported specific fault and cancel programming process
ECU-B Save current statement and close diagnostic 、Prohibit Send message while reading the state status till power off
Y
ECU-A PC client feedback can program and restart, enter boot mode start program
10S time out
No more Wait for a request to send，PC side timeout exit boot request
After ECU-A programmed finish，restart，enter the normal mode, the received VCU's 255STATE, enter 255 mode
Power on/off again, all the ECU enter into the normal power-up mode

VCU 控制程序刷写流程
VCU control program process:


The test start Successively:
0x7DF:02 10 03;
0x7C2:31 01 02 03
Other ECU
VCU confirmation request to their own program and send 7CA: 7F 31 78, requesting PC waits (10 times,once 1S)
VCU received and determine whether according to program state
Y
N
VCU enter 255STATE, and broadcast in all segments （5 times, each time interval of 10ms）
VCU ignore 7DF message
All ECU receives VCU's 255STATE, enter 255 mode (stop diagnosis, stop sending business messages)
Tester reported specific fault and cancel programming process