1 About this Document

This document specifies the behavior of the Bosch specific UDS Flash programming Sequence. Original is located in CDG Doors: https://rb-alm-06-p-dwa.de.bosch.com:8443/dwa/rm/urn:rational::1-3a1288eb3f4712b8-M-00168b20

2 Change History

2.1 Version 2.1

- NRC 33 removed from Service 0x36 and 0x37 to be complient to ISO 14229 from 2013

- NRC 24 removed from Service 0x34 to be complient to ISO 14229 from 2013

- Renaming of Headings.

- Version History added

2.2 Version 2.0

- Doors Export used for PDF generation

- Start Address und End Address replacing Block ID in Service 0x34 to support segments

- content of WDBI / RDBI Fingerprint made more generic

- MaxReprogValue added to RDBI ProgrammingCounter

2.3 Version 1.3:

- Services 0x28 and 0x85 only available in Extended Session

- Description of MaxInvalidCounter detailed

- NRC 0x22 added for Services 0x10, 0x11, 0x27, 0x2E, 0x31

- NRC 0x31 removed from Service 0x31 Check Program Dep.

- NRC 0x70 added to Service 0x34

- NRC 0x71 added to Service 0x36

- BlockState removed from RDBI / WDBI Fingerprint

- MaxReprogValue removed from RDBI ProgrammingCounter

2.4 Version 1.2:

- Replace Tables with Excelobjects → better Doors Im-/Export

2.5 Version 1.1

- Flow charts for diagnostic services added. The flow charts are

based on reference implementation.

2.6 Version 1.0:

- Initial Version based on the previous delivery for special OEMs

3 System Description

The bootblock provides the capability of exchanging the application software of the control unit via external tester tools by means of diagnostic services. Diagnostic communication provides services to:

- Identify the ECU
- Unlock the ECU security mechanism for authorized access
- Erase the existing application software
- Download a new application software or parameter sets.
- Verify the integrity and authenticity of the downloaded data

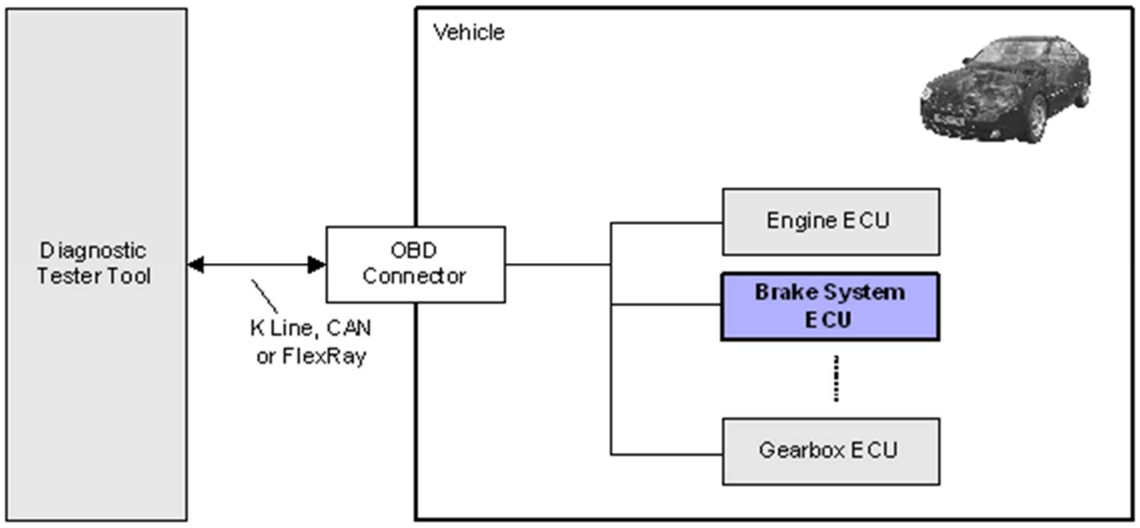
Diagnostic bootblock communication can be used for end-of-line reprogramming or in the servicing workshop. A safe and secure reprogramming can be provided by the bootblock.

This document describes the diagnostic communication functions and specifications of the flash Reprogramming Session.

4 Context information

4.1 Operational environment

Within a vehicle, normally several ECUs exist. These ECUs share a common diagnostic bus. For Bootblock Diagnostic Communication a Tester is attachable to the Bus to perform Diagnostic and Download Communication.



(example for Brake System ECU)

4.2 Use cases

The Diagnostic Tester Tool can be a production facility in the vehicle production plant or a tool that is used in the repair shop. These tester tools send diagnostic requests to the ECUs in the vehicle and evaluate the responses. As the brake system ECU is also part of the vehicle, all Bosch brake system ECU’s support a diagnosis communication interface to transfer information from an ECU to an off-board tester and vice-versa.

The information transferred consists of ECU fault information, signal reporting, control routines, calibration values and many other types of data used either in performing ECU diagnostics, re-programming or any other data transfer not associated with normal node to node communication.

The Bootblock as part of Diagnostic Communication has three major tasks:

1. Startup Consistency check, to evaluate if the application is valid and remain in secure bootmode if it is not.

2. Provide a safe reprogramming functionality, to exchange the application with different versions.

3. Manage the history of all downloaded SW-versions and types

4.3 Referenced Documents

4.3.1 International Documents

The general requirements of the diagnosis communication protocol are based on ISO requirements, specified in the following documents:

ISO 14229-1: Road Vehicles - Unified Diagnostic Services (UDS)

ISO 15765-2: Road Vehicles - Diagnostics on CAN: Network Layer Services

ISO 15765-3.3: Road Vehicles - Diagnostics on controller area network (CAN) - Part 3: Implementation of unified diagnostic services (UDS on CAN)

4.4 Abbreviations & Definitions

Abbreviation	Description
Bootblock	Independent Software which is permanently present in the ECU containing startup and SW-download functionality.
BootManager	Part of Bootblock, which performs SW-validity checks during start-up and starts either BootLoader or application SW.
BootLoader	Part of the Bootblock, which holds all functionality to erase, download and check application SW.
ECU	ElectronicControlUnit; In this document the word ECU is used for the device the bootblock running on.
SWIL	Software interlock is a part of the SW which is necessary to erase and modify the flash. This part of the SW will be downloaded separatly to RAM. For example this can be the Flashdriver.
Tester	External tool, which requests service execution on the ECU. Tester - ECU communication is a Client - Server relation.
UDS	UnifiedDiagnosticServices; Specified diagnostic protocol in ISO14429

5 Requirements

5.1 General Requirements

5.1.1 Safe Reprogramming

Definition: Safe Reprogramming is, when an ECU remains reprogrammable after power-down if the application is not valid.

5.1.2 Secure Reprogramming

The Bootblock has to ensure that the downloaded application is valid for this ECU. It shall not be possible to start any malicious code, changed by accident or by purpose.

5.1.3 Access Protection

The Bootloader has to protect the SW from unauthenticated manipulation.

5.1.4 Hardware Requirements

Bootblock has to ensure that the specified data-retention-time will not be compromised. This includes a limitation of FLASH erase and/or write cycles

5.2 System Behaviour

5.2.1 System Startup

After System Startup (hard / soft reset) a so called BootManager shall be executed.

The BootManager will determine if application (FSW) is valid or a programming request is present.

Based on these states execution will either continue in Application (FSW) or Bootloader (BLDR).

5.2.2 Diagostic Communication

Bootloader shall implement a Diagnostic Communication Interface to the vehicle bus.

Supported bus systems can be:

- CAN

- FlexRay

- Ethernet

- ...

The ECU is supporting Half-Duplex communication only. This means that

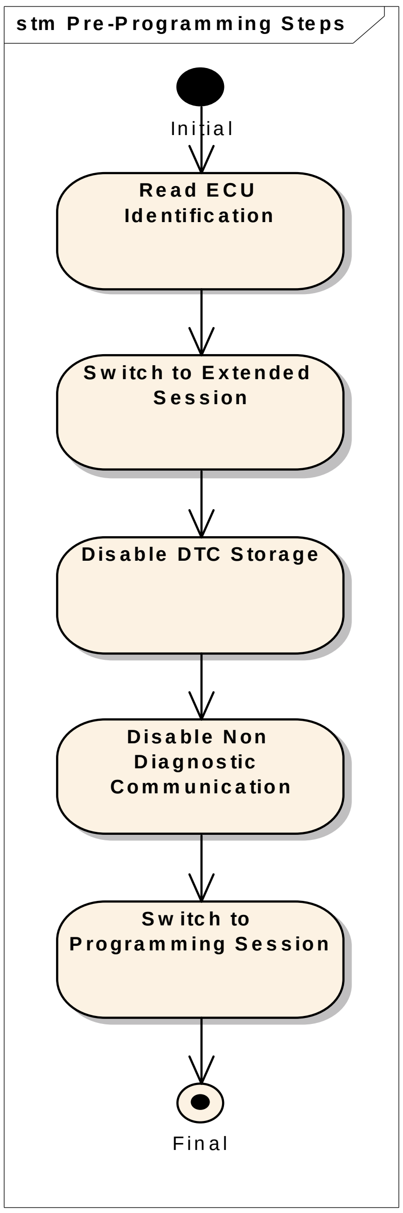
communication between a diagnostic tool and the ECU is only possible in one direction at the same time.

There is only one diagnostic protocol instance available in the ECU, which can handle only one request at a time. The rule is that any received message occupies this resource until the request message is processed.

5.2.3 Software Update

5.2.3.1 Pre-Programming Steps

Overview:


stm Pre-Programming Steps
Initial
Read ECU Identification
Switch to Extended Session
Disable DTC Storage
Disable Non Diagnostic Communication
Switch to Programming Session
Final

Read ECU Identification:

In this step the ECU will be identified. Hardware and Software information will be requested by the tester so that the test can check if the following download steps are allowed.

For doing this step, more Data IDs will be read out by sending 0x22 service to the ECU. The ECU has to respond with information of the ECU hardware and installed software

Switch to Extended Session:

The ECU shall switch to extended Mode

Disable DTC Storage:

Usually be sent functional. All ECUs connected to the bus shall stop storing trouble codes, as signals may be invalid or missing during download.

Disable Non Diagnostic Communication:

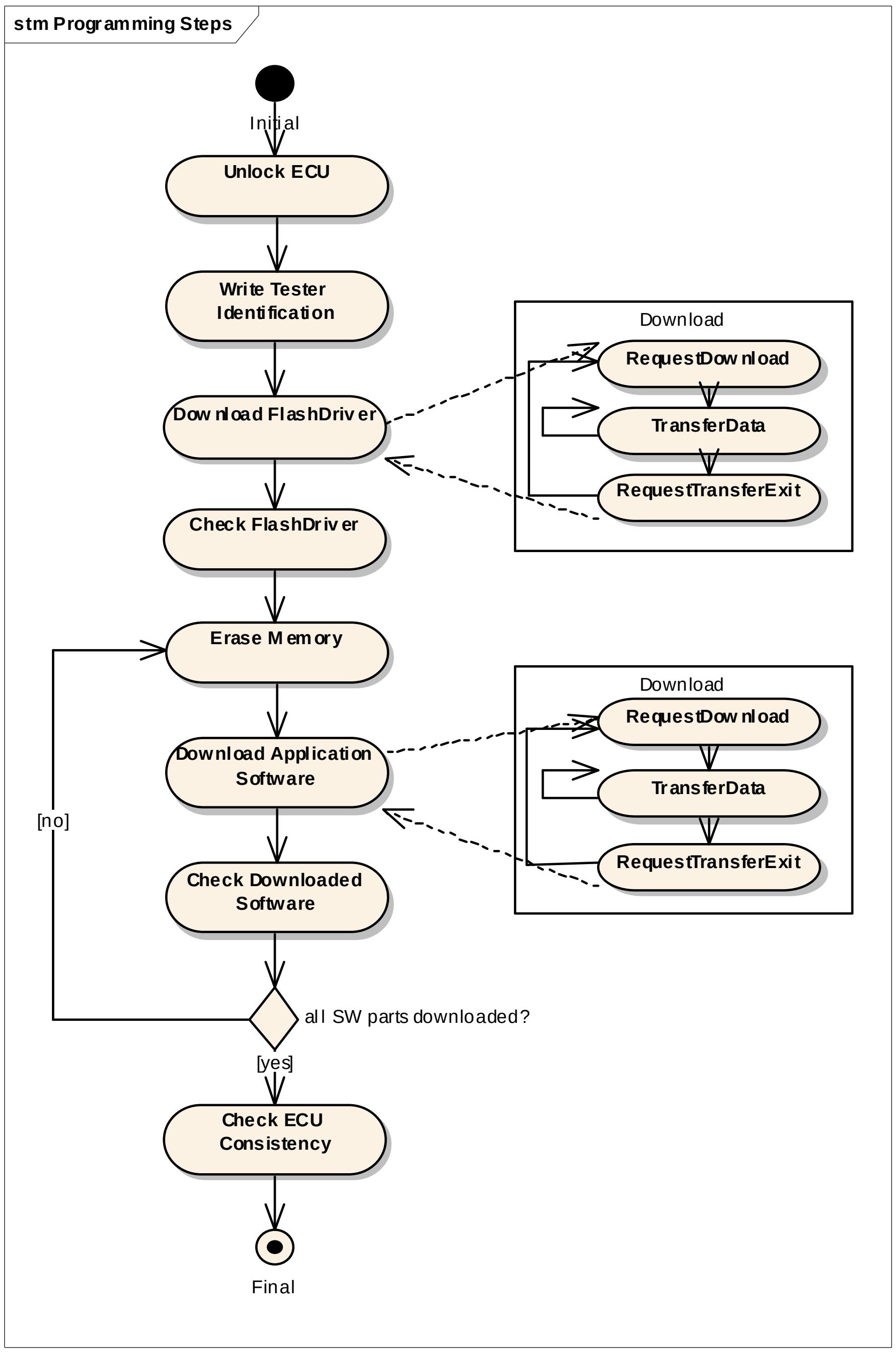
ECU shall stop sending messages other than required for SW-Download, to get a minimal bus load.

Switch to Programming Session:

If ECU is ready for Reprogramming the Tester shall send the Programming Session Request. The ECU will perform a reset and start in BootLoader mode.

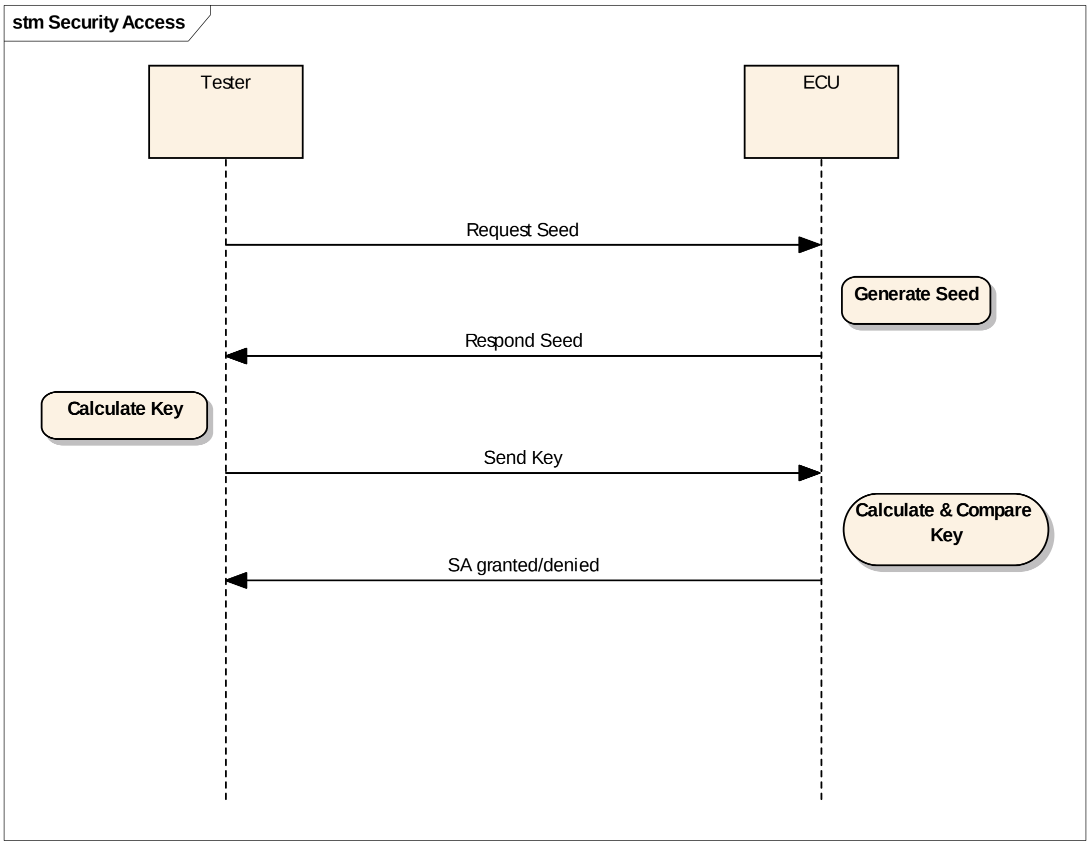
5.2.3.2 Programming Steps

Overview:



Unlock ECU:

To enable Reprogramming the tester has to authenticate itself and unlock the ECU.



Write Tester Identification:

The Tester has to submit its identification to the ECU.

[optinal] Download FlashDriver:

If the flashdriver do not reside in ROM the flashwrite and flasherase routines have to be downloaded to the ECU.

Erase Memory:

Using Flash technologies the ROM cannot be overwritten, so an explicit erase routine has to be performed.

Download Application Software:

During download ECU writes the transferred data into the predefined ROM area.

Check Downloaded Software:

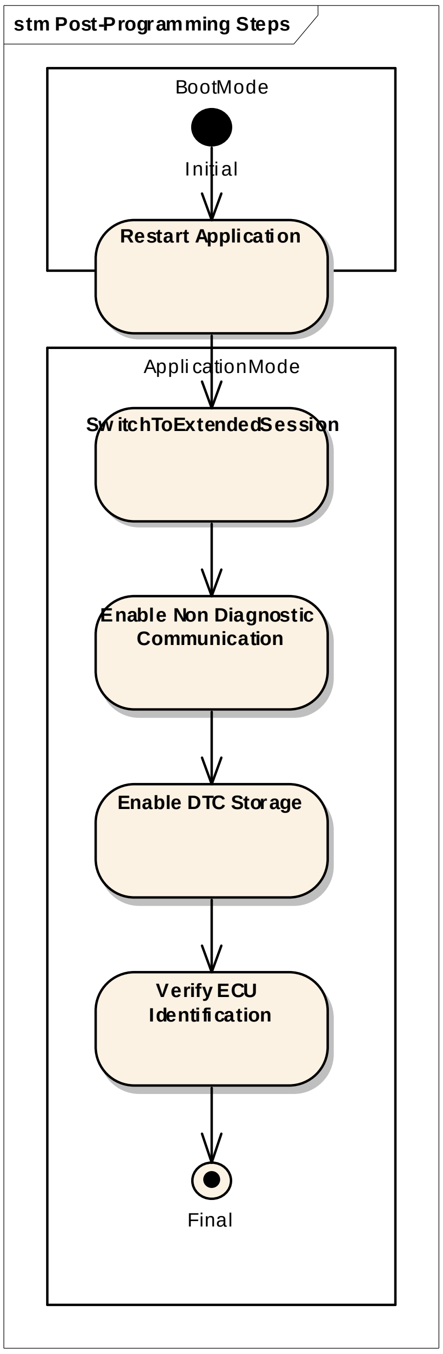
After transfer the ECU has to check if the written data are valid -> Secure Programming

Check ECU Consistency:

If all blocks have been downloaded the ECU has to check the consistency between the blocks. The validity state of the application software will be set here.

5.2.3.3 Post-Programming Steps

Overview:



Restart Application:

After Software download Tester has to restart the ECU, with an ECU-Reset request.

Switch to Extended Session:

The ECU shall switch to extended Mode.

Enable Non Diagnostic Communication:

To reach normal mode Tester has to enable normal communication on all ECUs.

Enable DTC Storage:

After normal communication is enabled the all error handlers can be restarted.

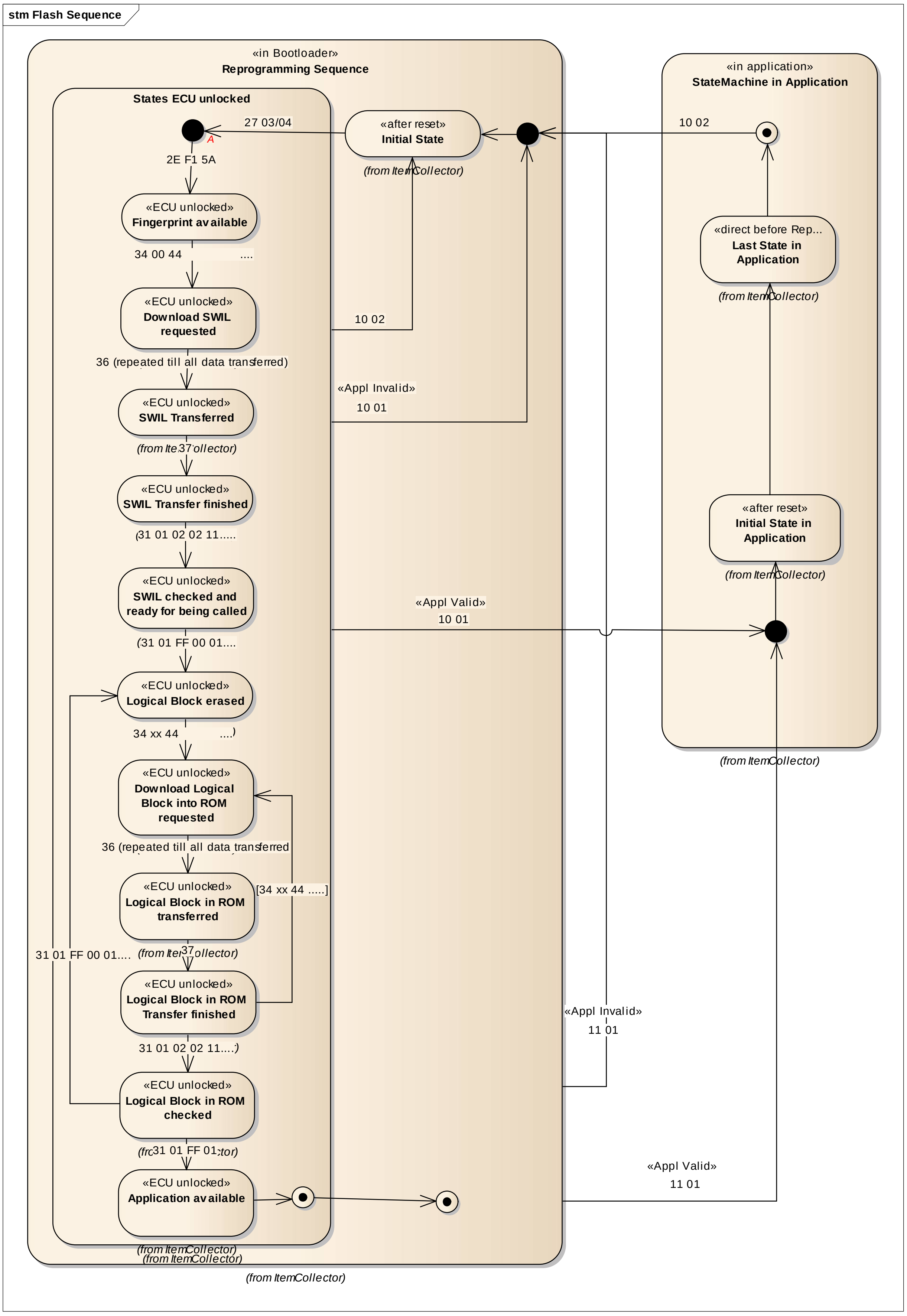
Verify ECU Identification:

Tester can check if desired software is present in ECU now.

5.2.3.4 Reprogramming Sequence Control

During reprogramming, the predefined sequence shall be followed strictly. As showed in the figure below, the Bootloader shall maintain a state machine according to each reprogramming step. This will ensure that the ECU will always be in a deterministic state.

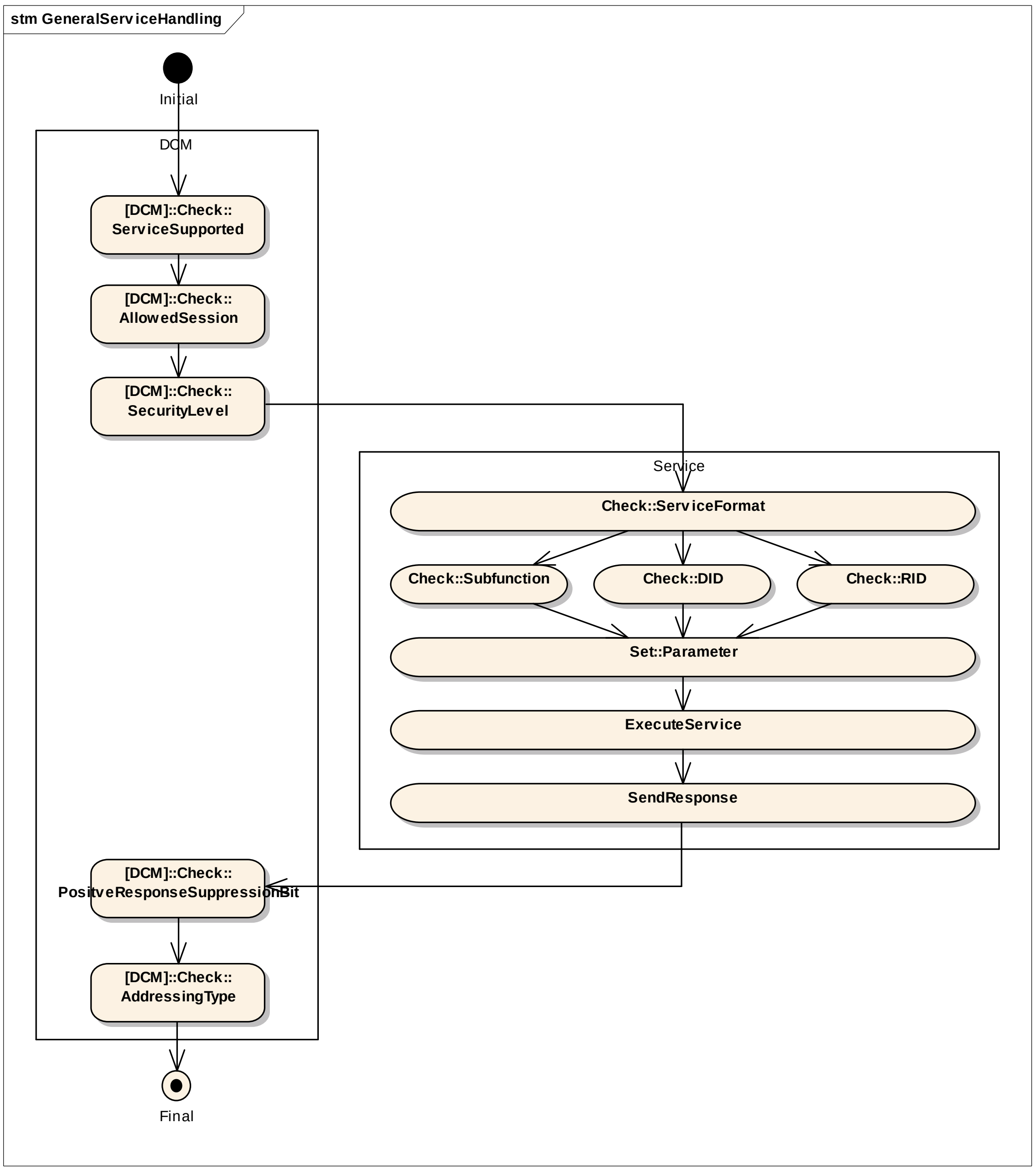
In the chart, all possible states of ECU are defined. Certain service request will cause a state transition. Only transitions which are depicted are allowed. That means, all the service requests, which cause state transitions, will be supported and served by the bootloader ONLY when the current state of ECU equals to the one at the tail side of the transition. Otherwise the bootloader will report a NRC with Request Sequence Error (0x24), and stop the service processing.


stm Flash Sequence
«in Bootloader»
Reprogramming Sequence
States ECU unlocked
27 03/04
A
2E F1 5A
«ECU unlocked»
Fingerprint available
34 00 44 ....
«ECU unlocked»
Download SWIL requested
36 (repeated till all data transferred)
«ECU unlocked»
SWIL Transferred
(from ItemCollector)
37
«ECU unlocked»
SWIL Transfer finished
(31 01 02 02 11.....
«ECU unlocked»
SWIL checked and ready for being called
(31 01 FF 00 01....
«ECU unlocked»
Logical Block erased
34 xx 44 ....)
«ECU unlocked»
Download Logical Block into ROM requested
36 (repeated till all data transferred
«ECU unlocked»
Logical Block in ROM transferred
[34 xx 44 .....]
31 01 FF 00 01....
(from ItemCollector)
37
«ECU unlocked»
Logical Block in ROM Transfer finished
31 01 02 02 11....)
«ECU unlocked»
Logical Block in ROM checked
(from ItemCollector)
31 01 FF 01
«ECU unlocked»
Application available
(from ItemCollector)
(from ItemCollector)
(from ItemCollector)
«after reset»
Initial State
(from ItemCollector)
10 02
«Appl Invalid»
10 01
«Appl Valid»
10 01
«Appl Invalid»
11 01
«Appl Valid»
11 01
«in application»
StateMachine in Application
10 02
«direct before Rep...
Last State in Application
(from ItemCollector)
«after reset»
Initial State in Application
(from ItemCollector)
(from ItemCollector)

5.3 Diagnostic Behaviour

5.3.1 General Diagnostic Behaviour

General Service Execution


stm GeneralServiceHandling
Initial
DCM
[DCM]::Check:: ServiceSupported
[DCM]::Check:: AllowedSession
[DCM]::Check:: SecurityLevel
Service
Check::ServiceFormat
Check::Subfunction
Check::DID
Check::RID
Set::Parameter
ExecuteService
SendResponse
[DCM]::Check:: PositveResponseSuppressionBit
[DCM]::Check:: AddressingType
Final

General Diagnostic behaviour shall be implemented according to ISO14429 UDS on CAN.

5.3.1.1 Negative Response Codes (NRC)

The following table specifies all allowed negative response codes:

NRC	NegativeResponse	Condition
10 H	General Reject	*The requested action has been rejected by the ECU. This reponse code shall only be implemented if none of the negative response code defined below meet the needs of the implementation*
11 H	ServiceNotSupported	*Reception of a non-wanted ServiceIdentifier*
12 H	SubFunctionNotSupported	*The sub-function of the service is not supported*
13 H	InvalidFormat	*Request format is not valid*
21 H	BusRepeatRequest	*ECU can not process the current request. A repetition of the service is expected.*
22 H	ConditionsNotCorrect	*Criteria of the Request are not met*
24 H	RequestSequenceError	*The service order does not correspond with the default.*
31 H	RequestoutOfRange	*The request contains invalid Data*
33 H	SecurityAccessDenied	*For the service, the ECU must be unlocked.*
35 H	invalidKey	*The key sent does not match the server's internally calculated key*
36 H	exceededNumberOfAttempt	*The requested action will not be taken because the tester has unsuccessfully attempted to gain security access more times than the ECU security strategy allows.*
37 H	requiredTimeDelayNotExpired	*Delay Timer is active and a request is transmitted*
70 H	uploadDownloadNotAccepted	*An attempt to upload/download can not be accomplished due to some fault conditions*
71 H	transferDataSuspended	*A data transfer operation has been halted due to some fault.*
72 H	generalProgrammingFailure	*This response code indicates that the server detected an error while erasing or programming a memory location in the permanent memory device (e.g. FlashMemory)*
73 H	wrongBlockSequenceCounter	*An error has been detected in the BlockSequenceCounter value.*
78 H	requestCorrectlyReceived-ResponsePending	*The request has been received correctly, and all the data are valid, but the action to be performed is not yet completed and ECU is not ready to receive another request.*
7E H	SubfunctionNotSupported-InActiveSession	*The sub-function of the request is not supported in the actual session.*
7F H	ServiceNotSupported-InActiveSession	*The request is not supported in the actual session.*

5.3.2 Diagnostic Session Model

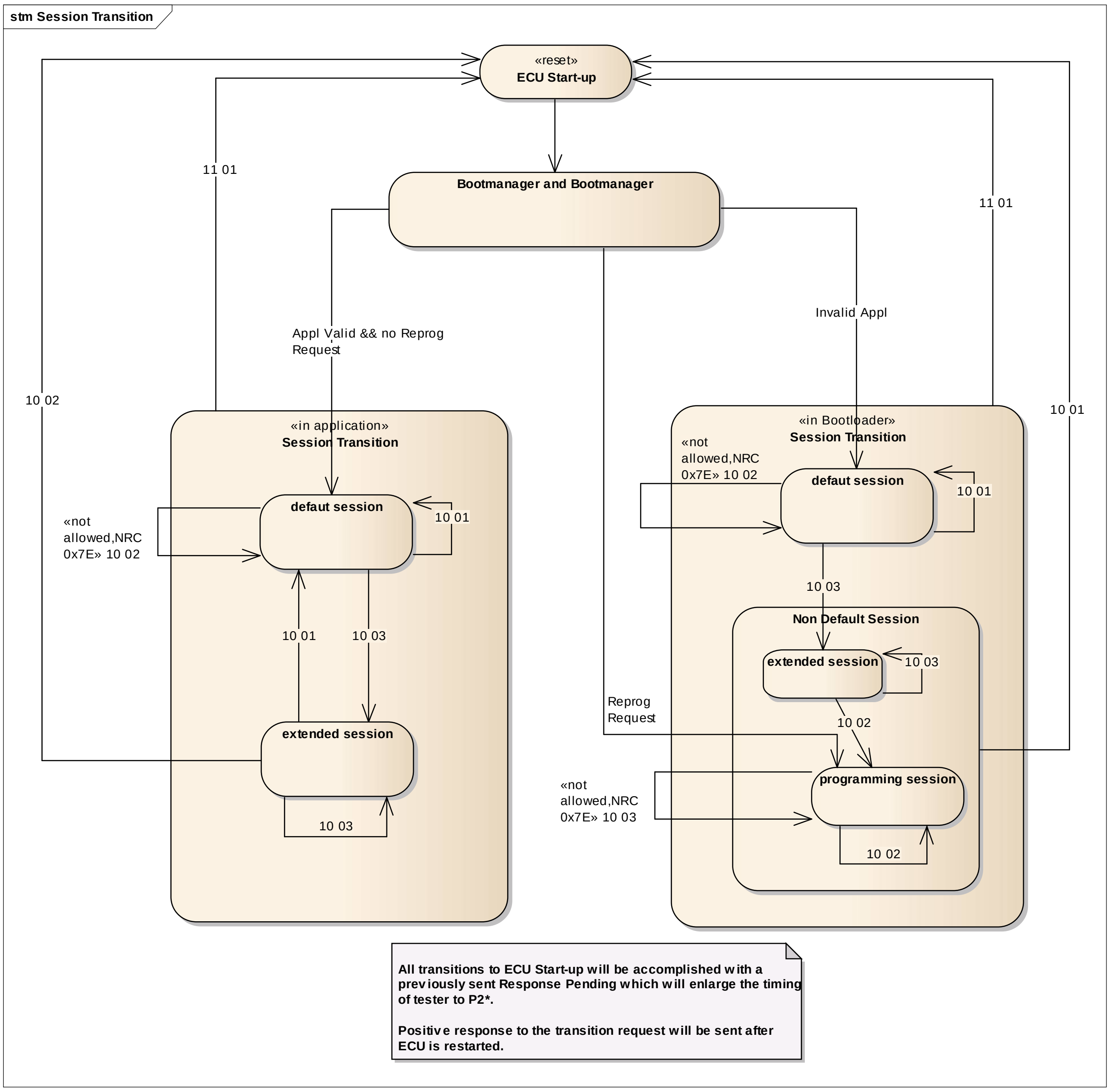
The following table shows in which session the UDS services are allowed to be executed:

	Default-Session	Extended-Session	Programming-Session	Addressing Mode*
DiagnosticSessionControl	x	x	x	p/f
ECUReset	x	x	x	p/f
ReadDataById	x	x	x	p/f
SecurityAccess			x	p
CommunicationControl		x		p/f
WriteDataById			x	p
RoutineControl				
RC::EraseMemory			x	p
RC::CheckMemory			x	p
RC::CheckProgrammingDependencies			x	p
RequestDownload			x	p
TransferData			x	p
RequestTransferExit			x	p
TesterPresent	x	x	x	p/f
ControlDTCSettings		x		p/f

*p: phyical addressing, f: functional addressing

5.3.3 Diagnostic Session Transition

The figure below depicts the session transitions and ECU reset behaviour:



All transitions to ECU Start-up will be accomplished with a previously sent Response Pending which will enlarge the timing of tester to P2*.

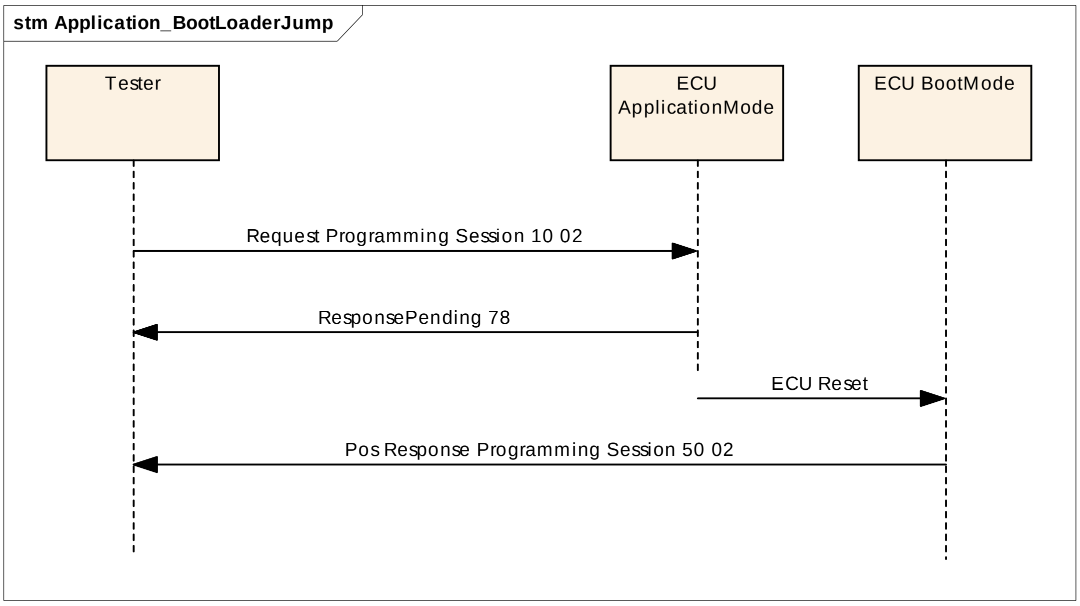
Positive response to the transition request will be sent after ECU is restarted.

5.3.3.1 Application - Bootloader Transition

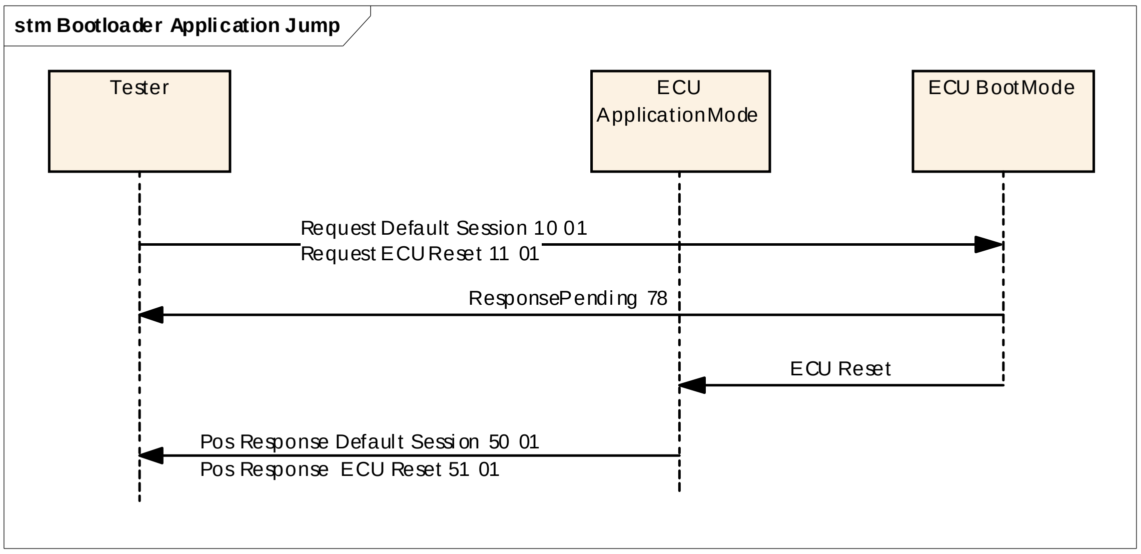
The mode from Application to Bootloader and back from Bootloader to Application may take more than typical serivce execution time, due to possible (re)initializations.

Therefore a Response Pending mechanism as described in ISO14229 shall be used.

Jump from Application to Bootloader:


stm Application_BootLoaderJump
Tester
ECU ApplicationMode
ECU BootMode
Request Programming Session 10 02
ResponsePending 78
ECU Reset
Pos Response Programming Session 50 02

Jump from Bootloader to Application:


stm Bootloader Application Jump
Tester
ECU ApplicationMode
ECU BootMode
Request Default Session 10 01
Request ECU Reset 11 01
ResponsePending 78
ECU Reset
Pos Response Default Session 50 01
Pos Response ECU Reset 51 01

5.3.4 Diagnostic Security Model

The following table shows the security protected services:

	Security Access required
DiagnosticSessionControl	
ECUReset	
ReadDataById	
SecurityAccess	
CommunicationControl	
WriteDataById	**x**
RoutineControl	
RC::EraseMemory	**x**
RC::CheckMemory	**x**
RC::CheckProgrammingDependencies	**x**
RequestDownload	**x**
TransferData	**(x)***
RequestTransferExit	**(x)***
TesterPresent	
ControlDTCSettings	

*: According to ISO 14229 NRC 33 is not available for Service TransferData and Request Transfer Exit. So Security Access Check is NOT done in this Services. Never the less both Services are only possible to start after Request Download was passed successfully including Security Check.

Request on services which are protected by Security Access Mechanisms will be rejected by the ECU as long as the security level has not been activated.

5.4 Supported Diagnostic Services

5.4.1 Diagnostic Session Control

5.4.1.1 Request

Byte 0	Service ID = 0x10
Byte 1	Target Session ID

Target Session ID	
0x01	Default Session
0x02	Programming Session
0x03	Extended Session

5.4.1.2 Positive Response

Byte 0	Service ID + 0x40 = 0x50
Byte 1	Target Session ID
Byte 2..3	P2Max Timing Parameter
Byte 4..5	P2*Max Timing Parameter

P2Max is the maximum time between request and first response in that session. The scale is 1ms.

P2*Max is the maximum response time if ECU has enlarged the time window by sending the ResponsePending message (7F SID 78) for delayed response. The scale is 10ms.

5.4.1.3 Negative Response

NRC::12: Unknown sessionId sent

NRC::13: Service has the wrong length

NRC::22: ResponsePending could not be sent or PDM operation fails.

NRC::7E: Subfunction not allowed in current session. Session transition is not allowed.

5.4.2 ECU Reset

5.4.2.1 Request

Byte 0	Service ID = 0x11
Byte 1	Reset Type, only hard reset 0x01 is supported by bootblock

5.4.2.2 Positive Response

Byte 0	Service ID + 0x40 = 0x51
Byte 1	Reset Type, only 0x01 supported

5.4.2.3 Negative Response

NRC::12: Unknown resetType sent.

NRC::13: Service has the wrong length.

NRC::22: ResponsePending could not be sent or PDM operation fails.

5.4.3 Read Data by Identifier

5.4.3.1 Request

Byte 0	Service ID = 0x22
Byte 1..2	Data ID

DataIdentifier (DID) indicate the data which shall be read from/written into the ECU:

DataID	**Description**	**Output Size (in Bytes)**
0xFD06	Active DiagnosticSession	1
0xF185	ProgrammingCounter	4
0xF15B	FingerPrint Read	30
0xF187	VehicleManufacturerPartNumber	16
0xF190	VehicleIdentificationNumber	17
0xF194	ApplicationSoftwareVersionNumber	4
0xF195	ApplicationSoftwareNumber	3
0xF196	BootblockSoftwareVersionNumber	3
0xF197	BootblockSoftwareNumber	3

5.4.3.2 Positive Response

Byte 0	Service ID + 0x40= 0x62
Byte 1..2	Data ID
Byte 3..N	Data, whose length N - 3 + 1, N >= 3 bytes

5.4.3.2.1 Active Diagnostic Session (0xFD06)

Byte 3..3	SessionID + 0x80

Session ID:

0x01	Default Session
0x03	Extended Session
0x02	Programming Session

The MSB should be set which indicates that the ECU is in bootloader mode.

Therefore the data field to this request is 0x81, 0x82, and 0x83.

5.4.3.2.2 Programming Counter (0xF185)

Byte (3)*N	Block ID
Byte (4..5)*N	ProgrammingAttemptCounter
Byte (6..7)*N	Max Programming Counter

N: number of logical blocks

A SWIL Block will not have a Programming counter and will not be included in the RDBI

5.4.3.2.3 Fingerprint Read (0xF15B)

Byte (3) * N	Block ID
Byte (4) * N	Block Status
Byte (5..13) * N	Fingerprint

N: number of logical blocks

A SWIL Block will not have a Fingerprint and will not be included in the RDBI

5.4.3.2.4 Vehicle Manufacturer Part Number (0xF187)

Byte 3..18	VehicleManufacturerPartNumber, defined by the customer project

5.4.3.2.5 Vehicle Identification Number (0xF190)

Byte 3..19	VehicleIdentificationNumber, , defined by the customer project

5.4.3.2.6 Appl Sofware Version Number (0xF194)

Byte 3..5	SoftwareVersionNumber, defined by the customer project

5.4.3.2.7 Appl Software Number (0xF195)

Byte 3..5	SoftwareNumber, defined by teh customer project

5.4.3.2.8 Bootblock Software Version Number (0xF196)

Byte 3	Year of current bootblock release in hex format, e.g. 0x0B means year 2011
Byte 4	Calender week of current bootblock release in hex format, e.g. 0x0B means CW11
Byte 5	Patch level of current bootblock release in hex format

5.4.3.2.9 Bootblock Software Number (0xF197)

Byte 3..5	Project spcific number. Will be defined project/OEM specifically.

5.4.3.3 Negative Response

NRC::13: Service has the wrong length.

NRC::22: PDM operation fails.

NRC::31: Sent DataID is unknown.

5.4.4 Security Access

The Security Access Service (SA) has to provide a locking mechanism to deny unauthenticated access.

The sequence according to Security Model has to be executed.

5.4.4.1 Request

Byte 0	Service ID = 0x27
Byte 1	SubFunction
[Byte 2..5]	SA-Parameter

SubFunction	
0x03	RequestSeed
0x04	SendKey

SA-Parameter	
RequestSeed	- empty -
SendKey	Key for unlocking the ECU

5.4.4.2 Positive Response

Request Seed:

Byte 0	Service ID + 0x40 = 0x67
Byte 1	0x03
Byte 2..5	Seed

Send Key:

Byte 0	Service ID + 0x40 = 0x67
Byte 1	0x04

In case the ECU is already unlocked the response to RequestSeed shall be 0x00 0x00 0x00 0x00

In Case seed is requested several Times without sending a Key, the same Seed shall be returned

Definition of the Security Access Statemachine including TimeLock and AttemptCounter:

Description:

- TimeLock will be defined as **10 seconds**.
- MaxInvalidCounter is set to **3**
- MaxInvalidCounter shall be strored in NvM
- If MaxInvalidCounter reaches Maximum the TimeLock shall be activated on every Start of Boot after Reset.
- MaxInvalidCounter will be reset if a valid Key was received.
- The Security Algorithm is defined in an extra document.

5.4.4.3 Negative Response

NRC::12: Unknown Subfunction sent.

NRC::13: Service has the wrong length.

NRC::22: ResponsePending could not be sent or PDM operation fails.

NRC::24: Request sequence error.

NRC::35: Wrong Key.

NRC::36: NumberOfAttemptsExceeded.

NRC::37: TimeoutNotExpired.

5.4.5 Communication Control

5.4.5.1 Request

Byte 0	Service ID = 0x28
Byte 1	ControlType
Byte 2	CommunicationType

ControlType	
0x00	enableRxAndTx
0x01	enableRxAndDisableTx

CommunicationType	
0x01	NormalCommunicationMessages

5.4.5.2 Positive Response

Byte 0	Service ID + 0x40 = 0x68
Byte 1	ControlType

5.4.5.3 Negative Response

NRC::12: Unknown ControlType sent.

NRC::13: Service has the wrong length.

NRC::31: Unknown CommunicationType sent.

NRC::7F: Service not supported in active session.

5.4.6 Write Data by Identifier

Write Data By Identifier Service (WDBID) shall be used to store data in the ECU non-volatile memory.

5.4.6.1 Request Write Fingerprint (0xF15A)

Byte 0	Service ID = 0x2E
Byte 1	0xF1
Byte 2	0x5A
Byte 3..11	TesterFingerPrint

TesterFingerprint:

TesterFingerPrint	
Byte 3..11	Fingerprint

The Fingerprint is stored in a RAM Variable at first. During Erase this fingerprint is stored into the NvM ID corresponding to the erased Block.

5.4.6.2 Positive Response

Byte 0	Service ID + 0x40 = 0x6E
Byte 1..2	Data ID

5.4.6.3 Negative Response

NRC::13: Service has the wrong length.

NRC::22: ResponsePending could not be sent or PDM operation fails.

NRC::24: Request sequence error.

NRC::31: Sent DataID is unknown.

NRC::33: Security Access is denied.

5.4.7 Routine Control

5.4.7.1 Request

Byte 0	Service ID = 0x31
Byte 1	RoutienControlType, only StartRoutine (0x01 supported)
Byte 2..3	RoutineID
Byte 4..N	RoutineData

Routine Type	
0x01	StartRoutine
0x02	Stoproutine (Not supported)
0x03	RequestRoutineResults (Not supported)

RoutineID	
0xFF00	EraseMemory
0xFF01	CheckProgrammingDependencies
0x0202	CheckMemory

5.4.7.2 Positive Response

Byte 0	Service ID + 0x40 = 0x71
Byte 1	RoutienControlType
Byte 2..3	RoutineID
Byte 4..N	RoutineResult

5.4.7.3 RoutineID Definition

5.4.7.3.1 Erase Memory

The routine for EraseMemory is used to erase parts of the Flash-ROM.

5.4.7.3.1.1 Request

Byte 0	Service ID = 0x31
Byte 1	0x01
Byte 2..3	0xFF00
Byte 4	AddressNLengthFormatID
Byte 5	BlockIdentifictation

AddressNLengthFormatID	
0x01	Size of BlockId in Request (1Byte)

BlockId	
0x00 - 0xFF	BlockId

5.4.7.3.1.2 Functionality

The RC Service Erase Memory erases the ROM space of the given memory block.

The function has to increase the flashcounter on each erase attempt.

The state of the logical block has to be set to invalid.

The Fingerprint tranmitted with WDBI before Erase needs to be stored in the NvM ID configured for the Block to be erased

5.4.7.3.1.3 Positive Response

Byte 0	Service ID + 0x40 = 0x71
Byte 1	0x01
Byte 2..3	0xFF00
Byte 4	RoutineResult

RoutineResult	
0x01	Erase Failed
0x00	Erase Succeeded

5.4.7.3.1.4 Negative Response

NRC::22: ResponsePending could not be sent or PDM operation fails.

NRC::24: Request sequence error.

NRC::31: AddressNLengthFormatID is not valid. Logical Block not found.

NRC::33: Security Access is denied.

5.4.7.3.2 Check Memory

The Check Memory routine shall check that the downloaded block has been transferred without manipulation (biterrors,...). This is part of the "secure" download concept.

5.4.7.3.2.1 Request

Byte 0	Service ID = 0x31
Byte 1	0x01
Byte 2..3	0x0202
Byte 4	AddressLengthFormatID
Byte 5	BlockID (alway one byte)
Byte 6..(6+LengthFormat-1)	Length of Checksum
Byte (6+LengthFormat)..(6+LengthFormat+Length of Checksum Value -1)	ChecksumField

AddressNLengthFormatID	
Bit 7..4	Size of Checksum Length Field (0: no checksum transmitted)
Bit 3..0	Size of BlockID in Request = 1

BlockId	
0x00 - 0xFF	BlockId

Length of Checksum	
Byte number according to LengthFormat	Hex value indicates Size of Checksum

5.4.7.3.2.2 Functionality

The bootloader shall calculate the checksum of the downloaded data. This checksum shall be compare to the precalculated checksum the tester is sending with the request. That way a fault free transmission from tester to ECU can be guaranteed.

CheckSum Algorithm → CRC32

Additionally a precalculated checksum shall be located at the end of the download data which is written into flash ROM during download phase. This Checksum shall also be calculated by the Boot and compared to the one in ROM. That way a manipulation in the downloaded Data can be detected. The position as well as the usage of this ROM Checksum shall be configurable.

CheckSum Algorithm → CRC32

If both Checksums match the downloaded data will be set to valid.

5.4.7.3.2.3 Positive Response

Byte 0	Service ID * 0x40 = 0x71
Byte 1	0x01
Byte 2..3	0x0202
Byte 4	RoutineResult

RoutineResult	
0x01	Check Failed
0x00	Check Succeeded

5.4.7.3.2.4 Negative Response

NRC::22: Response Pending can not be sent.

NRC::24: Request sequence error.

NRC::31: Logical Block not found.

NRC::33: Security Access is denied.

5.4.7.3.3 Check Programming Dependencies

The check programming dependencies service shall check the consistency between the blocks.

5.4.7.3.3.1 Request

Byte 0	Service ID = 0x31
Byte 1	0x01
Byte 2..3	0xFF01

5.4.7.3.3.2 Functionality

Bootblock will update the blockstatus information of all logical blocks according to the result of check memory.

5.4.7.3.3.3 Positive Response

Byte 0	Service ID + 0x40 = 0x71
Byte 1	0x01
Byte 2..3	0xFF01
Byte 4	RoutineResult

RoutineResult	
0x01	CheckProgrammingDependencies Failed
0x00	CheckProgrammingDependencies Succeeded

5.4.7.3.3.4 Negative Response

NRC::22: Response Pending can not be sent.

NRC::24: Request Sequence Error.

NRC::33: Security Access is denied.

5.4.8 Request Download

The Request Download Service (RD) shall prepare the parameter / variables for the download.

5.4.8.1 Request

Byte 0	Servicer ID = 0x34
Byte 1	DataFormatID
Byte 2	Address&LengthFormatID
Byte 3..6	Download Start Address
Byte 7..10	Download Length

DataFormatID	
0x00	Unencrypted / UnCompressed
0x01	Unencrypted / LZ77 for Uncompression (rba_Bldr_Compression needed)

Address&LengthFormatID	
Bit 7..4	Size of Download Length in Request, (4 is recommended)
Bit 3..0	Size of Download Start Address in Request (4 is recommended)

Start Address	Start Address of Download

Download Length	Size of data which shall be downloaded with this Request

In case of compression used the size will be the compressed size.

The DownloadLength has to be smaller or equal to the length of the reserved logical block.

5.4.8.2 Positive Response

Byte 0	Service ID + 0x40 = 0x74
Byte 1	LengthFormatID
Byte 2..N	MaxBlockLength

LengthFormatID	
Bit 7..4	Size of length information in response message, is fixed to 2
Bit 3..0_	Reserved, always 0

MaxBlockLen	The number of bytes which can be transferred in one transfer data service

N = 2 + Size of length information - 1

5.4.8.3 Negative Response

NRC::22: DataFormatId invalid or AddressNLengthFormatID invalid.

NRC::31: Logical Block not found.

NRC::33: Security Access is denied.

NRC::70: Block was not Erased before

5.4.9 Transfer Data

The Transfer Data Service (TD) is used to transfer and store the new application code into the ECU.

5.4.9.1 Request

Byte 0	Service ID = 0x36
Byte 1	BlockSequenceNumber, Block means data unit which shall be transferred by one Transfer Data Request
Byte 2..N	Data

Definition BlockSequenceNumber:

- One byte, range 0x00 – 0xFF
- Initvalue is: 0x01
- increased in every transfer data request
- overrun at 0xFF (next value is 0x00)

5.4.9.2 Positive Response

Byte 0	Service ID + 0x40 = 0x76
Byte 1	BlockSequenceNumber, Block means data unit which shall be transferred by one Transfer Data Request

5.4.9.3 Negative Response

NRC::13: Request Length is not correct.

NRC::22: Unable to send ResponsePending.

NRC::24: Request Sequence Error.

NRC::31: writing the Data would exceed the logical block area (defined by RequestDownload)

NRC::71: No more retries allowed for this Sequence number

NRC::72: Unable to write the data to the Flash-ROM.

NRC::73: Received BlockSequenceCounter does not match the expected one. If the last data block is repeated by the tester with the last processed sequence number, it will be accepted with a postive response. But no write operation is performed by the bootloader.

5.4.10 Request Transfer Exit

The Request Transfer Exit Service (TE) is used to signal the end of a download.

5.4.10.1 Request

Byte 0	Service ID = 0x37

5.4.10.2 Positive Response

Byte 0	Service ID + 0x40 = 0x77

5.4.10.3 Negative Response

NRC::13: Request format is not correct.

NRC::24: Request Sequence Error.

5.4.11 Tester Present

The Tester Present Service (TP) is used as a keep alive information to keep the bootblock from triggering the S3 timeout.

5.4.11.1 Request

Byte 0	Service ID = 0x3E
Byte 1	0x00

5.4.11.2 Positive Response

Byte 0	Service ID + 0x40 = 0x7E
Byte 1	0x00

5.4.11.3 Negative Response

NRC::12: Subfunction not supported.

NRC::13: Request format is not correct.

5.4.12 Control DTC Settings

The Control DTC Settings Service (CDS) is used to toggle the storage of diagnostic trouble codes. In bootblock this is only a dummy implementation.

5.4.12.1 Request

Byte 0	Service ID = 0x85
Byte 1	DTCSettingType

5.4.12.2 Positive Response

Byte 0	Service ID + 0x40 = 0xC5
Byte 1	DTCSettingType

DTCSettingType	
0x01	On
0x02	Off

5.4.12.3 Negative Response

NRC::12: Subfunction not supported.

NRC::13: Request format is not correct.