Great Wall

GWMLAN General Specification

Flash BootLoader Specification

Author	Zhoujiancang
Approver	Liubin

Confidentiality
Information in this document is the sole property of GWM and must not be disclosed to any third party without the prior written permission from GWM.

Revision History
Revision history describes the changes in the new issue. Please refer to the release notes.

Contents

1 Introduction 1
1.1 Overview 1
1.2 Abbreviation 1
1.3 Document References 2
2 Diagnostic Communication 3
2.1 CAN Bus Requirements 3
2.2 LIN Bus Requirements 3
3 Diagnostic Protocol 4
3.1 UDS Services 4
3.2 Data Identifiers 6
3.3 Routines 11
RS-FBL-3110 Erase Flash Memory (0xFD00) 11
RS-FBL-3120 Check Programming Dependency (0xFD01) 12
RS-FBL-3130 Check FBL PreCondition (0xFD02) 12
RS-FBL-3140 Check Flash Driver Validity (0xFD03) 13
RS-FBL-3150 Check AppSwAppDataValidity (0xFD04) 14
4 FBL Architecture 16
4.1 ECU UDS on CAN and LIN Protocol Stack for FBL 16
4.2 ECU Side System Architecture for FBL 16
4.3 Reprogramming Sequence 17
4.4 ECU Programming Detailed Scenarios Steps 21
4.4.1 ECU Programming Control flow 27
4.4.2 ECU Initialization 27
4.4.3 S3Server Timer Check 32
4.4.4 Reprogramming with Flash Driver Download 33
4.4.5 ECU Reset 39
4.4.6 Reprogramming Initialization 40
5 FBL Requirements 44
5.1 System Requirements for reprogrammable ECUs 44
5.2 ECUs System Requirements for FBL Implementation 45
6 Checksum Algorithm 46

1 Introduction

This document defines the requirements of BootLoader software and programming process.

1.1 Overview

This specification documentation defines the system requirements for the flashing reprogramming of vehicle ECU by using UDS protocol PDU based on CAN bus and LIN bus. The supporting diagnostic network communication requirements are defined. The Flash boot loader system architecture and the downloading related requirements are also described in detail.

1.2 Abbreviation

Abbreviation	Description
CAN	Controller Area Network
DID	Data identifier
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read Only Memory
ID	Identifier
ISO	International Organization for Standardization
N_Ar	Network layer timing parameter Ar
N_As	Network layer timing parameter As
N_Br	Network layer timing parameter Br
N_Bs	Timeout Flow Control
N_Cs	Time until transmission of next Consecutive Frame
N_Cr	Timeout Consecutive Frame
NRC	Negative Response Code
OSI	Open Systems Interconnection
RAM	Random Access Memory
ROM	Read Only Memory
STmin	Minimum Separation Time
SID	Service Identifier
SW	Software
UDS	Unified Diagnostic Service

1.3 Document References

[1] ISO 14229-1:2013 Road Vehicles -- Unified Diagnostic Services (UDS) -- Part 1: Specification and requirements.
[2] ISO 15765-2: 2016 Road vehicles -- Diagnostic communication over Controller Area Network (DoCAN) -- Part 2: Transport protocol and network layer services.
[3] ISO 14229-3:2012 Road Vehicles -- Unified Diagnostic Services (UDS) -- Part 3: Unified diagnostic services on CAN implementation (UDSonCAN).
[4] ISO 11898-1:2015 Road vehicles -- Controller area network (CAN) - Part 1: Data link layer and physical signaling.
[5] ISO 11898-2:2016 Road vehicles -- Controller area network (CAN) - Part 2: High-speed medium access unit.
[6] ISO 15031-4:2014 Road vehicles -- Communication between vehicle and external equipment for emissions-related diagnostics -- Part 4: External test equipment.
[7] ISO 15031-7:2013 Road vehicles -- Communication between vehicle and external equipment for emissions-related diagnostics -- Part 7: Data link security
[8] GWMLAN00-09 CAN Diag Req Spec
[9] GWMLAN00-20 LIN Diag Req Spec
[10] GWMLAN00-17 Security_Algorithm Spec

2 Diagnostic Communication

This section defines the reprogramming specific requirements regarding the diagnostic communication on CAN and LIN networks. The general diagnostic requirements are specified in GWMLAN Diagnostic Requirement Specification. The used protocol is the Unified Diagnostic Services Protocol according to ISO 15765.

2.1 CAN Bus Requirements

RS-FBL-2010 Application layer timing parameter for CAN bus
The Application layer timing parameters shall be applied as defined in requirement **RS-DS-4120** in REF [8].

RS-FBL-2020 TP timing parameter for CAN bus
The TP timing parameters shall be applied as defined in requirements **RS-DS-4040** and **RS-DS-4050** in REF [8].

2.2 LIN Bus Requirements

RS-FBL-2030 Application layer timing parameter for LIN bus
The Application layer timing parameters shall be applied as defined in requirement **RS-DS-4020** in REF [9].

RS-FBL-2040 TP timing parameter for LIN bus
The TP timing parameters shall be applied as defined in requirement **RS-DS-3050** in REF [9].

RS-FBL-2050 TP frames type in LIN bus
Apply as specified in chapter 3.2 in REF [9].

RS-FBL-2060 The LIN diagnostic class III must be supported
Apply as specified in chapter 2.1 in REF [9].

RS-FBL-2070 LIN configuration and identification should not be used in booting mode
Any requirements related to the LIN node dynamic configuration and identification can only be implemented in the application run time mode, no LIN configuration and identification should be supported after the FBL booting mode is entered.

3 Diagnostic Protocol

The GWM generic Flash bootloader system uses the Unified Diagnostic Services (UDS) based on CAN bus and LIN bus to implement the application software downloading. Of course, except the flashing on-line downloading, the UDS on CAN or LIN protocol stack can also support other diagnostic functionalities, such as configuration, calibration, and DTC managing, in this FBL system requirement specification, will focus on the diagnostic services.

3.1 UDS Services

RS-FBL-3010 UDS services supported in the programmable ECUs

For all of ECUs, the below UDS services and sub-functions must be supported. In the below table in which defines the UDS services must be supported in the application for FBL feature, the 1st column describe the UDS service which is needed to support, the 2nd column describe the respective sub-function which is need to supported; the 3rd and 4th columns to define if this service and sub-function should be supported in the physical addressing mode or functional addressing mode respectively; the columns from 5th to 7th defines , if the current active diagnostic session mode is default mode , programming mode, or extended mode , this service and the relating sub-function can be supported or not.

Table 1: UDS Services

Svc	S-Func/ Param	Phy Addr	Func Addr	Default	Programming	Extended
0x10	0x01	yes	yes	yes	yes	yes
0x10	0x02	yes	no	no	no	yes
0x10	0x03	yes	yes	yes	no	yes
0x11	0x01	yes	yes	yes	yes	yes
0x11	0x03	yes	yes	yes	yes	yes
0x22	DID	yes	yes	yes	yes	yes
0x27	01	yes	no	no	no	yes
0x27	02	yes	no	no	no	yes
0x27	35	yes	no	no	yes	no
0x27	36	yes	no	no	yes	no
0x28	0x03 / 0x83 0x01	yes	yes	no	no	yes
0x28	0x00 / 0x80 0x01	yes	yes	no	no	yes
0x2E	DID	yes	no	no	yes	yes
0x31	RID	yes	no	no	yes	yes
0x34	Params	yes	no	no	yes	no

0x36	Params	yes	no	no	yes	no
0x37	N/A	yes	no	no	yes	no
0x3E	0x00/0x80	yes	yes	yes	yes	yes
0x85	0x01/0x81	yes	yes	no	no	yes
0x85	0x02/0x82	yes	yes	no	no	yes

RS-FBL-3020 Deviation from UDS (REF [1])

Please note: Table 1 restricts the addressing types of single services (e.g. 0x10 0x02) to physical only. The UDS specification defines that all services can be addressed functional.

If the tester sends an “invalid” functional request, the ECU shall ignore this request.

RS-FBL-3030 UDS Services

The detailed definition of the UDS services is specified in chapter 5 in REF [8].

RS-FBL-3040 Which type of the reset should be used to switching between the application software running mode and the boot software running mode

For all of the ECUs which is supporting the FBL downloading features, when the software running mode is switched from FBL boot software running mode into the application running mode, should use the 0x11 01 hard reset, i.e. the initialization result of reset should be same as the power on reset. If the ECU supporting the FBL can not support the 0x11 01 hard reset, but only 0x11 03 soft reset, then this must be discussed with GWM, and allowed by GWM. When the FBL downloading sequence is finished, and the boot software mode will be exit, then the type of 0x11 01 hard reset must be used, then after this hard reset, the application will get the CPU to run.

RS-FBL-3050 0x27 security access service used in the application software and the FBL boot software

The below sub functions for the requesting seeds and the submitting keys must be supported : 0x01, 0x02, 0x35, 0x36. All of the other sub functions can be added to support in the application software and FBL boot software, after the discussion between GWM and the ECU supplier. The 0x27 service must have the protective mechanism to define the repetitive spiteful then a mandatory delay timer will be started, before it is expired, no new attempt is allowed.

- As for the ECUs which need four bytes key, the security algorithm please refers to REF [10]; and the security algorithm mask please refers to ECU Level Diagnostic Requirement..
- As for the ECUs which need sixteen bytes key, the security algorithm is AES-128[1]; and the security algorithm mask please refers to [8].

3.2 Data Identifiers

RS-FBL-3060 Data Identifier "ApplicationSoftwareProgrammingSuccessCounter" (0xF1AA)

Data	Name	Description	Cvt.	Value
#1	Request SID	ReadDataByIdentifier	M	22
#2	dataIdentifier [byte#1]	ApplicationSoftwareProgrammingSuccessCounter [byte#1]	M	F1
#3	dataIdentifier [byte#2]	ApplicationSoftwareProgrammingSuccessCounter [byte#2]	M	AA

Data	Name	Description	Cvt.	Value
#1	Response SID	ReadDataByIdentifier	M	62
#2	dataIdentifier [byte#1]	ApplicationSoftwareProgrammingSuccessCounter [byte#1]	M	F1
#3	dataIdentifier [byte#2]	ApplicationSoftwareProgrammingSuccessCounter [byte#2]	M	AA
#4	dataRecord [byte#1]	AppSwProgrammingSuccessCntr [byte#1]	M	00 - FF
#5	dataRecord [byte#2]	AppSwProgrammingSuccessCntr [byte#2]	M	00 - FF
#6	dataRecord [byte#3]	AppSwProgrammingSuccessCntr [byte#3]	M	00 - FF
#7	dataRecord [byte#4]	AppSwProgrammingSuccessCntr [byte#4]	M	00 - FF

RS-FBL-3070 Data Identifier "ApplicationSoftwareProgrammingAttemptCounter" (0xF1A5)

Data	Name	Description	Cvt.	Value
#1	Request SID	ReadDataByIdentifier	M	22
#2	dataIdentifier [byte#1]	ApplicationSoftwareProgrammingAttemptCounter [byte#1]	M	F1
#3	dataIdentifier [byte#2]	ApplicationSoftwareProgrammingAttemptCounter [byte#2]	M	A5

Data	Name	Description	Cvt.	Value
#1	Response SID	ReadDataByIdentifier	M	62
#2	dataIdentifier [byte#1]	ApplicationSoftwareProgrammingAttemptCounter [byte#1]	M	F1
#3	dataIdentifier [byte#2]	ApplicationSoftwareProgrammingAttemptCounter [byte#2]	M	A5

#4	dataRecord [byte#1]	AppSwProgrammingAttemptCntr [byte#1]	M	00 - FF
#5	dataRecord [byte#2]	AppSwProgrammingAttemptCntr [byte#2]	M	00 - FF
#6	dataRecord [byte#3]	AppSwProgrammingAttemptCntr [byte#3]	M	00 - FF
#7	dataRecord [byte#4]	AppSwProgrammingAttemptCntr [byte#4]	M	00 - FF

RS-FBL-3080 Data Identifier "MaxAllowableProgrammingTimes" (0xF1A9)

Data	Name	Description	Cvt.	Value
#1	Request SID	ReadDataByIdentifier	M	22
#2	dataIdentifier [byte#1]	MaxAllowableProgrammingTimes [byte#1]	M	F1
#3	dataIdentifier [byte#2]	MaxAllowableProgrammingTimes [byte#2]	M	A9

Data	Name	Description	Cvt.	Value
#1	Response SID	ReadDataByIdentifier	M	62
#2	dataIdentifier [byte#1]	MaxAllowableProgrammingTimes [byte#1]	M	F1
#3	dataIdentifier [byte#2]	MaxAllowableProgrammingTimes [byte#2]	M	A9
#4	dataRecord [byte#1]	maxAllowableProgramNumber [byte#1]	M	00 - FF
#5	dataRecord [byte#2]	maxAllowableProgramNumber [byte#2]	M	00 - FF
#6	dataRecord [byte#3]	maxAllowableProgramNumber [byte#3]	M	00 - FF
#7	dataRecord [byte#4]	maxAllowableProgramNumber [byte#4]	M	00 - FF

RS-FBL-3090 ReadDataByIdentifier "FingerPrint" (0xF1F0)

Data	Name	Description	Cvt.	Value
#1	Request SID	ReadDataByIdentifier	M	22
#2	dataIdentifier [byte#1]	FingerPrint [byte#1]	M	F1
#3	dataIdentifier [byte#2]	FingerPrint [byte#2]	M	F0

Data	Name	Description	Cvt.	Value
#1	Response SID	ReadDataByIdentifier	M	62
#2	dataIdentifier [byte#1]	FingerPrint [byte#1]	M	F1
#3	dataIdentifier [byte#2]	FingerPrint [byte#2]	M	F0
#4	dataRecord [byte#3]	testSupplierId [byte#1]	M	00 - FF
#5	dataRecord [byte#4]	testSupplierId [byte#2]	M	00 - FF
#6	dataRecord [byte#5]	testSupplierId [byte#3]	M	00 - FF
#7	dataRecord [byte#6]	testSupplierId [byte#4]	M	00 - FF
#8	dataRecord [byte#7]	testerSerialNumber [byte#1]	M	00 - FF
#9	dataRecord [byte#8]	testerSerialNumber [byte#2]	M	00 - FF
#10	dataRecord [byte#9]	testerSerialNumber [byte#3]	M	00 - FF
#11	dataRecord [byte#10]	testerSerialNumber [byte#4]	M	00 - FF
#12	dataRecord [byte#11]	testerSerialNumber [byte#5]	M	00 - FF
#13	dataRecord [byte#12]	testerSerialNumber [byte#6]	M	00 - FF
#14	dataRecord [byte#13]	yearOffset (BCD)	M	00 - 99
#15	dataRecord [byte#14]	month (BCD)	M	01 – 12
#16	dataRecord [byte#15]	day (BCD)	M	01 – 31
#17	dataRecord [byte#16]	hour (BCD) (24 hours format)	M	00 – 23
#18	dataRecord [byte#17]	minute (BCD)	M	00 – 59
#19	dataRecord [byte#18]	seconds (BCD)	M	00 – 59
#20	dataRecord	shopOrGarageCoding (ASCII) [byte#1]	M	00 - FF

	[byte#19]			
#21	dataRecord [byte#20]	shopOrGarageCoding (ASCII) [byte#2]	M	00 - FF
#22	dataRecord [byte#21]	shopOrGarageCoding (ASCII) [byte#3]	M	00 - FF
#23	dataRecord [byte#22]	shopOrGarageCoding (ASCII) [byte#4]	M	00 – FF
#24	dataRecord [byte#23]	supplierSpecificFingerPrintData [byte#1]	M	00 – FF
#25	dataRecord [byte#24]	supplierSpecificFingerPrintData [byte#2]	M	00 – FF
#26	dataRecord [byte#25]	supplierSpecificFingerPrintData [byte#3]	M	00 – FF
#27	dataRecord [byte#26]	supplierSpecificFingerPrintData [byte#4]	M	00 – FF
#28	dataRecord [byte#27]	supplierSpecificFingerPrintData [byte#5]	M	00 – FF
#29	dataRecord [byte#28]	supplierSpecificFingerPrintData [byte#6]	M	00 – FF

RS-FBL-3100 WriteDataByIdentifier “FingerPrint” (0xF1F0)

Data	Name	Description	Cvt.	Value
#1	Request SID	WriteDataByIdentifier	M	2E
#2	dataIdentifier [byte#1]	FingerPrint [byte#1]	M	F1
#3	dataIdentifier [byte#2]	FingerPrint [byte#2]	M	F0
#4	dataRecord [byte#1]	testSupplierId [byte#1]	M	00 - FF
#5	dataRecord [byte#2]	testSupplierId [byte#2]	M	00 - FF
#6	dataRecord [byte#3]	testSupplierId [byte#3]	M	00 - FF
#7	dataRecord [byte#4]	testSupplierId [byte#4]	M	00 - FF
#8	dataRecord [byte#5]	testerSerialNumber [byte#1]	M	00 - FF
#9	dataRecord [byte#6]	testerSerialNumber [byte#2]	M	00 - FF
#10	dataRecord	testerSerialNumber [byte#3]	M	00 - FF

	[byte#7]			
#11	dataRecord [byte#8]	testerSerialNumber [byte#4]	M	00 - FF
#12	dataRecord [byte#9]	testerSerialNumber [byte#5]	M	00 - FF
#13	dataRecord [byte#10]	testerSerialNumber [byte#6]	M	00 - FF
#14	dataRecord [byte#11]	yearOffset (BCD)	M	00 - 99
#15	dataRecord [byte#12]	month (BCD)	M	01 – 12
#16	dataRecord [byte#13]	day (BCD)	M	01 – 31
#17	dataRecord [byte#14]	hour (BCD) (24 hours format)	M	00 – 23
#18	dataRecord [byte#15]	minute (BCD)	M	00 – 59
#19	dataRecord [byte#16]	seconds (BCD)	M	00 – 59
#20	dataRecord [byte#17]	shopOrGarageCoding (ASCII) [byte#1]	M	00 - FF
#21	dataRecord [byte#18]	shopOrGarageCoding (ASCII) [byte#2]	M	00 - FF
#22	dataRecord [byte#19]	shopOrGarageCoding (ASCII) [byte#3]	M	00 - FF
#23	dataRecord [byte#20]	shopOrGarageCoding (ASCII) [byte#4]	M	00 – FF
#24	dataRecord [byte#21]	supplierSpecificFingerPrintData [byte#1]	M	00 – FF
#25	dataRecord [byte#22]	supplierSpecificFingerPrintData [byte#2]	M	00 – FF
#26	dataRecord [byte#23]	supplierSpecificFingerPrintData [byte#3]	M	00 – FF
#27	dataRecord [byte#24]	supplierSpecificFingerPrintData [byte#4]	M	00 – FF
#28	dataRecord [byte#25]	supplierSpecificFingerPrintData [byte#5]	M	00 – FF
#29	dataRecord [byte#26]	supplierSpecificFingerPrintData [byte#6]	M	00 – FF

Data	Name	Description	Cvt.	Value
#1	Response SID	WriteDataByIdentifier	M	6E
#2	dataIdentifier [byte#1]	FingerPrint [byte#1]	M	F1
#3	dataIdentifier [byte#2]	FingerPrint [byte#2]	M	F0

3.3 Routines

RS-FBL-3110 Erase Flash Memory (0xFD00)

Data	Name	Description	Cvt.	Value
#1	Request SID	RoutineControl	M	31
#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	eraseMemory [byte#1]	M	FD
#4	routineIdentifier [byte#2]	eraseMemory [byte#2]	M	00
#5	addressAndLenghtFormat Identifier	44 means the length of address is 4 bytes, and the length of size is also 4 bytes	M	44
#6	routineControlOption [byte#1]	address [byte#1]	O	00-FF
#7	routineControlOption [byte#2]	address [byte#2]	O	00-FF
#8	routineControlOption [byte#3]	address [byte#3]	O	00-FF
#9	routineControlOption [byte#4]	address [byte#4]	O	00-FF
#10	routineControlOption [byte#1]	size [byte#1]	O	00-FF
#11	routineControlOption [byte#2]	size [byte#2]	O	00-FF
#12	routineControlOption [byte#3]	size [byte#3]	O	00-FF
#13	routineControlOption [byte#4]	size [byte#4]	O	00-FF

Data	Name	Description	Cvt.	Value
#1	Response SID	RoutineControl	M	71
#2	routineControlType	startRoutine	M	01

#3	routineIdentifier [byte#1]	eraseMemory [byte#1]	M	FD
#4	routineIdentifier [byte#2]	eraseMemory [byte#2]	M	00
#5	routineStatus	routineResult	M	00 - 01

Value (hex)	Description routineResult	Cvt.
0x00	Erase successful	M
0x01	Erase failure	M

RS-FBL-3120 Check Programming Dependency (0xFD01)

If only a single application software image is used, then always return TRUE.

Data	Name	Description	Cvt.	Value
#1	Request SID	RoutineControl	M	31
#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	checkProgramminDependency [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkProgramminDependency [byte#2]	M	01

Data	Name	Description	Cvt.	Value
#1	Response SID	RoutineControl	M	71
#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	checkProgrammingDependency [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkProgrammingDependency [byte#2]	M	01
#5	routineStatus	routineResult	M	00 – 01

Value (hex)	Description routineResult	Cvt.
0x00	correct result	M
0x01	incorrect result	M

RS-FBL-3130 Check FBL PreCondition (0xFD02)

Used to check if the environment conditions are acceptable or not before to execute actual programming actions.

Data	Name	Description	Cvt.	Value
#1	Request SID	RoutineControl	M	31

#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	checkFBLPreCondition [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkFBLPreCondition [byte#2]	M	02

Data	Name	Description	Cvt.	Value
#1	Response SID	RoutineControl	M	71
#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	checkFBLPreCondition [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkFBLPreCondition [byte#2]	M	02
#5	routineStatus	routineResult	M	00 – 01

Value (hex)	Description routineResult	Cvt.
0x00	check successful	M
0x01	check failed	M

RS-FBL-3140 Check Flash Driver Validity (0xFD03)

Data	Name	Description	Cvt.	Value
#1	Request SID	RoutineControl	M	31
#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	checkFlashDriverValidity [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkFlashDriverValidity [byte#2]	M	03
#5	routineControlOption [byte#1]	checksum [byte#1]	M	00-FF
#6	routineControlOption [byte#2]	checksum [byte#2]	M	00-FF
#7	routineControlOption [byte#3]	checksum [byte#3]	M	00-FF
#8	routineControlOption [byte#4]	checksum [byte#4]	M	00-FF

Data	Name	Description	Cvt.	Value
#1	Response SID	RoutineControl	M	71
#2	routineControlType	startRoutine	M	01

#3	routineIdentifier [byte#1]	checkFlashDriverValidity [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkFlashDriverValidity [byte#2]	M	03
#5	routineStatus	routineResult	M	00 – 01

Value (hex)	Description routineResult	Cvt.
0x00	check successful	M
0x01	check failed	M

RS-FBL-3150 Check AppSwAppDataValidity (0xFD04)

Used to check if the application software and data image is valid after it is downloaded into the flash memory of ECU.

Data	Name	Description	Cvt.	Value
#1	Request SID	RoutineControl	M	31
#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	checkAppSwAppDataValidity [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkAppSwAppDataValidity [byte#2]	M	04
#5	addressAndLenghtFormat Identifier	44 means the length of address is 4 bytes, and the length of size is also 4 bytes	M	44
#6	routineControlOption [byte#1]	FlashAddToCheck [byte#1]	M	00-FF
#7	routineControlOption [byte#2]	FlashAddToCheck [byte#2]	M	00-FF
#8	routineControlOption [byte#3]	FlashAddToCheck [byte#3]	M	00-FF
#9	routineControlOption [byte#4]	FlashAddToCheck [byte#4]	M	00-FF
#10	routineControlOption [byte#5]	FlashSizeToCheck [byte#1]	M	00-FF
#11	routineControlOption [byte#6]	FlashSizeToCheck [byte#2]	M	00-FF
#12	routineControlOption [byte#7]	FlashSizeToCheck [byte#3]	M	00-FF
#13	routineControlOption [byte#8]	FlashSizeToCheck [byte#4]	M	00-FF
#14	routineControlOption [byte#9]	checksum [byte#1]	M	00-FF

#15	routineControlOption [byte#10]	checksum [byte#2]	M	00-FF
#16	routineControlOption [byte#11]	checksum [byte#3]	M	00-FF
#17	routineControlOption [byte#12]	checksum [byte#4]	M	00-FF

Data	Name	Description	Cvt.	Value
#1	Response SID	RoutineControl	M	71
#2	routineControlType	startRoutine	M	01
#3	routineIdentifier [byte#1]	checkAppSwAppDataValidity [byte#1]	M	FD
#4	routineIdentifier [byte#2]	checkAppSwAppDataValidity [byte#2]	M	04
#5	routineStatus	routineResult	M	00 – 01

Value (hex)	Description routineResult	Cvt.
0x00	check successful	M
0x01	check failed	M

4 FBL Architecture

4.1 ECU UDS on CAN and LIN Protocol Stack for FBL

The GWM generic Flash boot loader system uses the Unified Diagnostic Services (UDS) based on CAN bus and LIN bus to implement the application software downloading. Of course, except the flashing on-line downloading, the UDS on CAN or LIN protocol stack can also support other diagnostic functionalities, such as configuration, calibration, and DTC managing, in this FBL system requirement specification, will focus on the diagnostic services.

4.2 ECU Side System Architecture for FBL

At ECU server side, the GWM flash boot loader system software architecture is based on the layered software architecture to design. The whole of UDS protocol stacks include the below sub modules: UDS session manager, diagnostic generic request/response manager, diagnostic services processing handler groups, generic protocol data management handler, and interfaces manager. In the below figure, only the FBL using diagnostic services are described.
The layered software design is applied to separate the hardware dependent low layer driver software and the generic reusable MCU and ECU modules service , such as LIN TP services, generic EEPROM or Emulated EEPROM services, generic watchdog timer services, and generic flash operation services, to make sure the whole of boot software design can be reusable as much as possible.
The FBL manager module is dedicated for Flash boot loading state managing, security related data managing, and downloading interfaces mapping

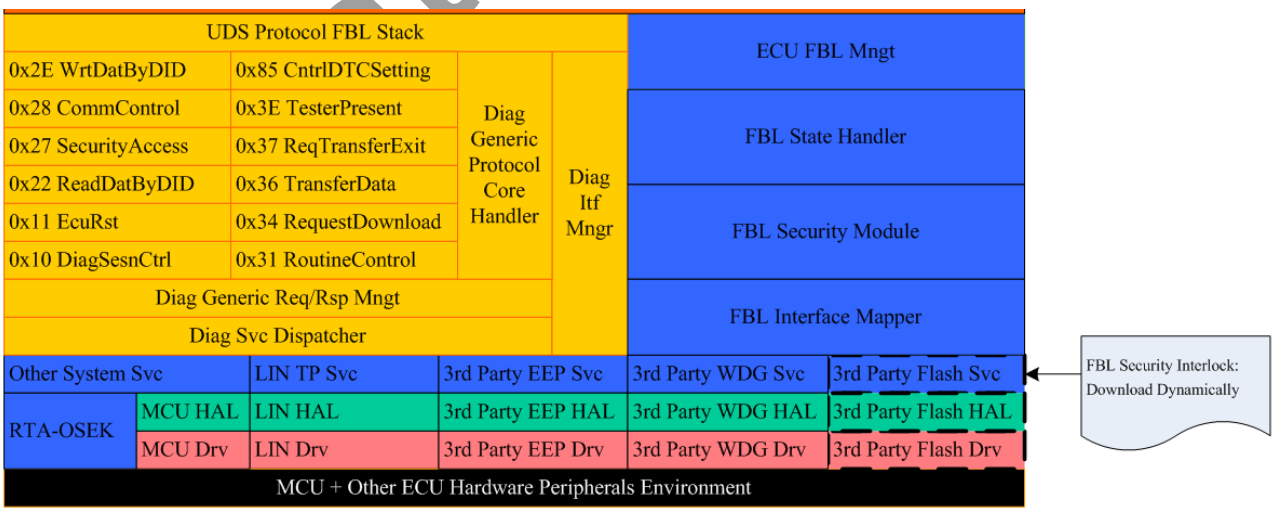

Figure 1: ECU side FBL architectural system compositions.

4.3 Reprogramming Sequence

The following section defines the reprogramming sequence based on the diagnostic communication between the tester and the ECU. In general, the sequence contains 4 parts:

1.) Pre-Programming Phase
2.) Flash Driver Download
3.) Programming Sequence
4.) Post-Programming Sequence

act SWDL

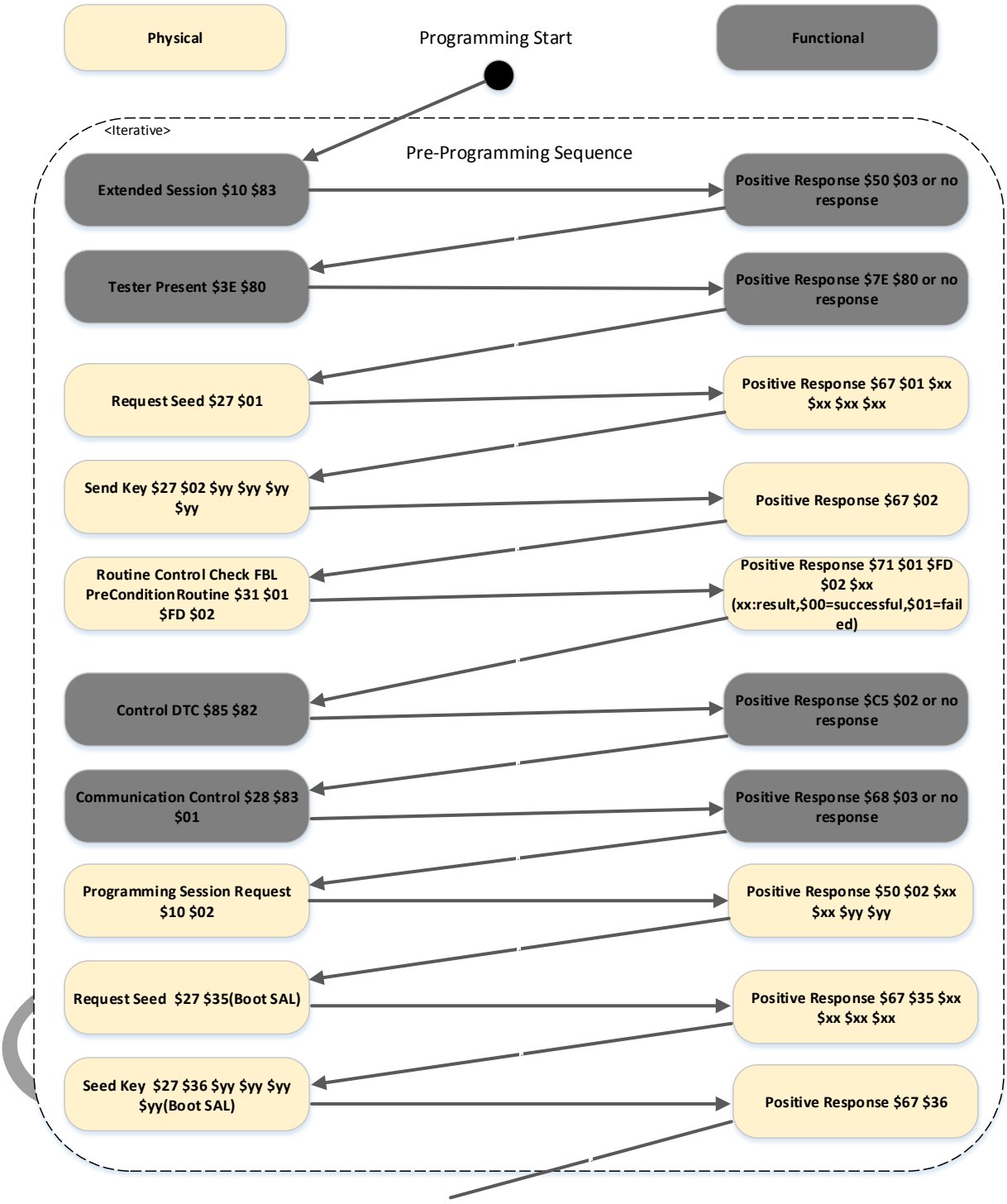

Figure 2: Pre-Programming sequence

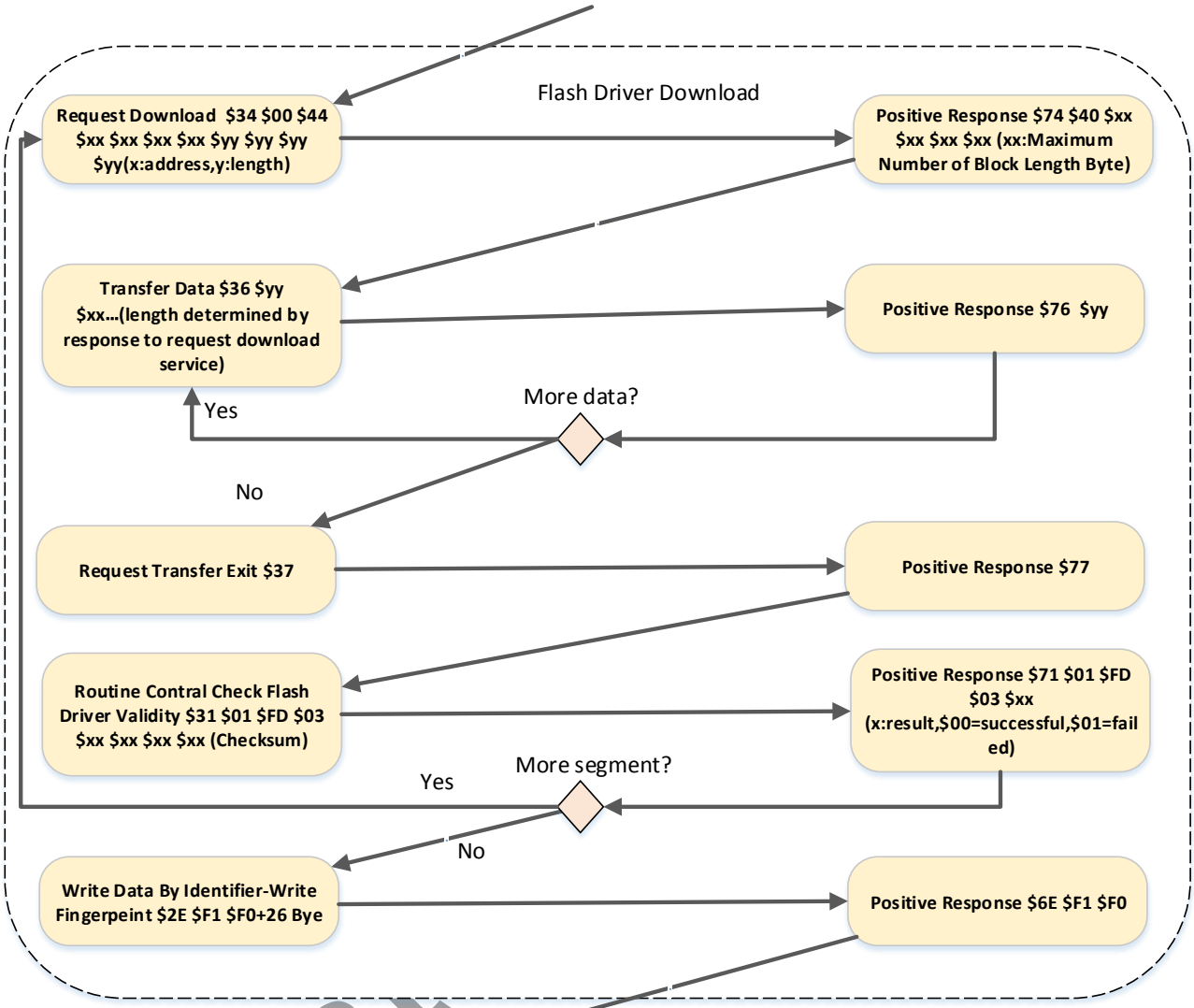

Figure 3: Flash Driver Download

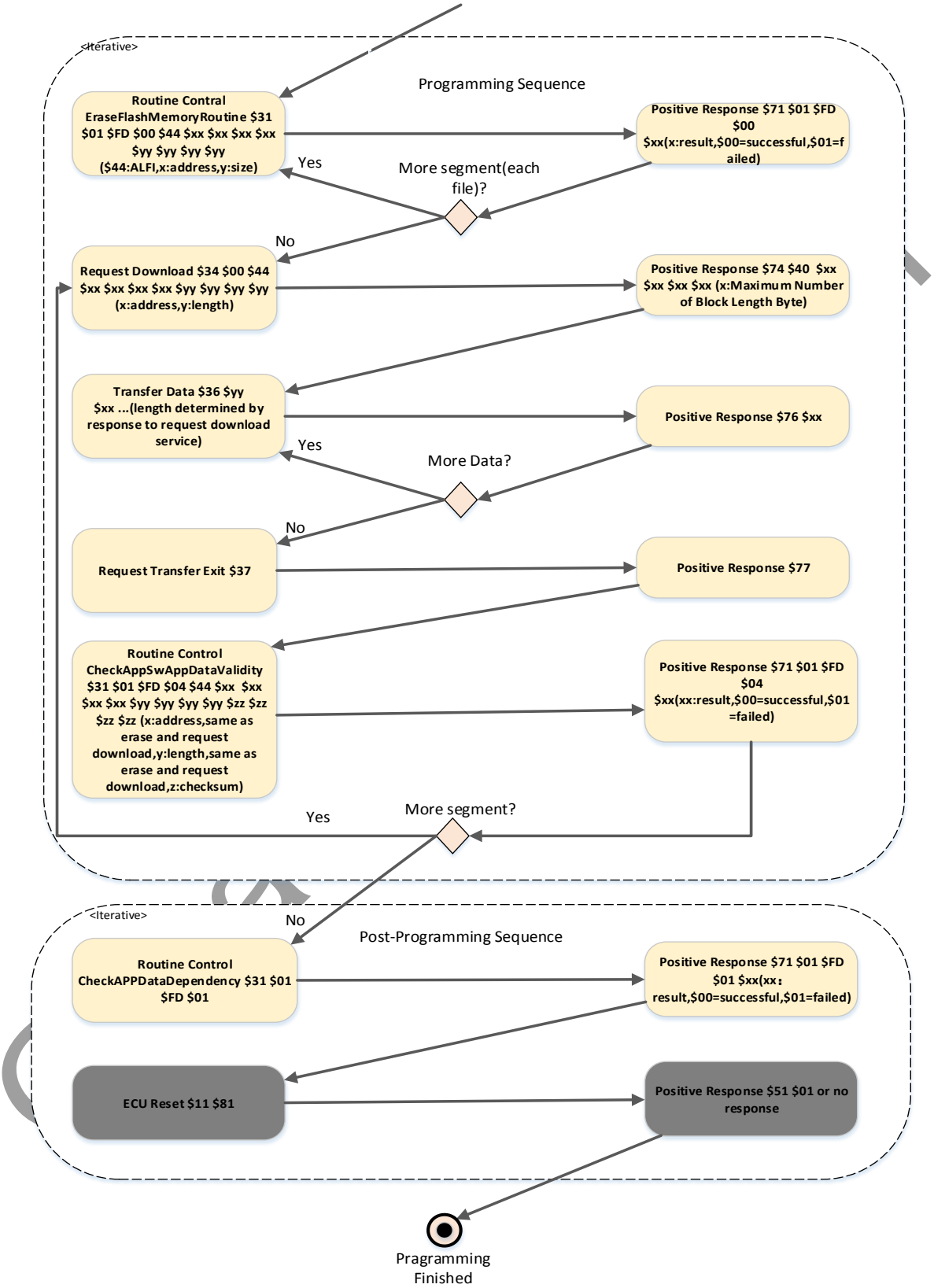

Figure 4: Programming and Post-Programming sequence

4.4 ECU Programming Detailed Scenarios Steps

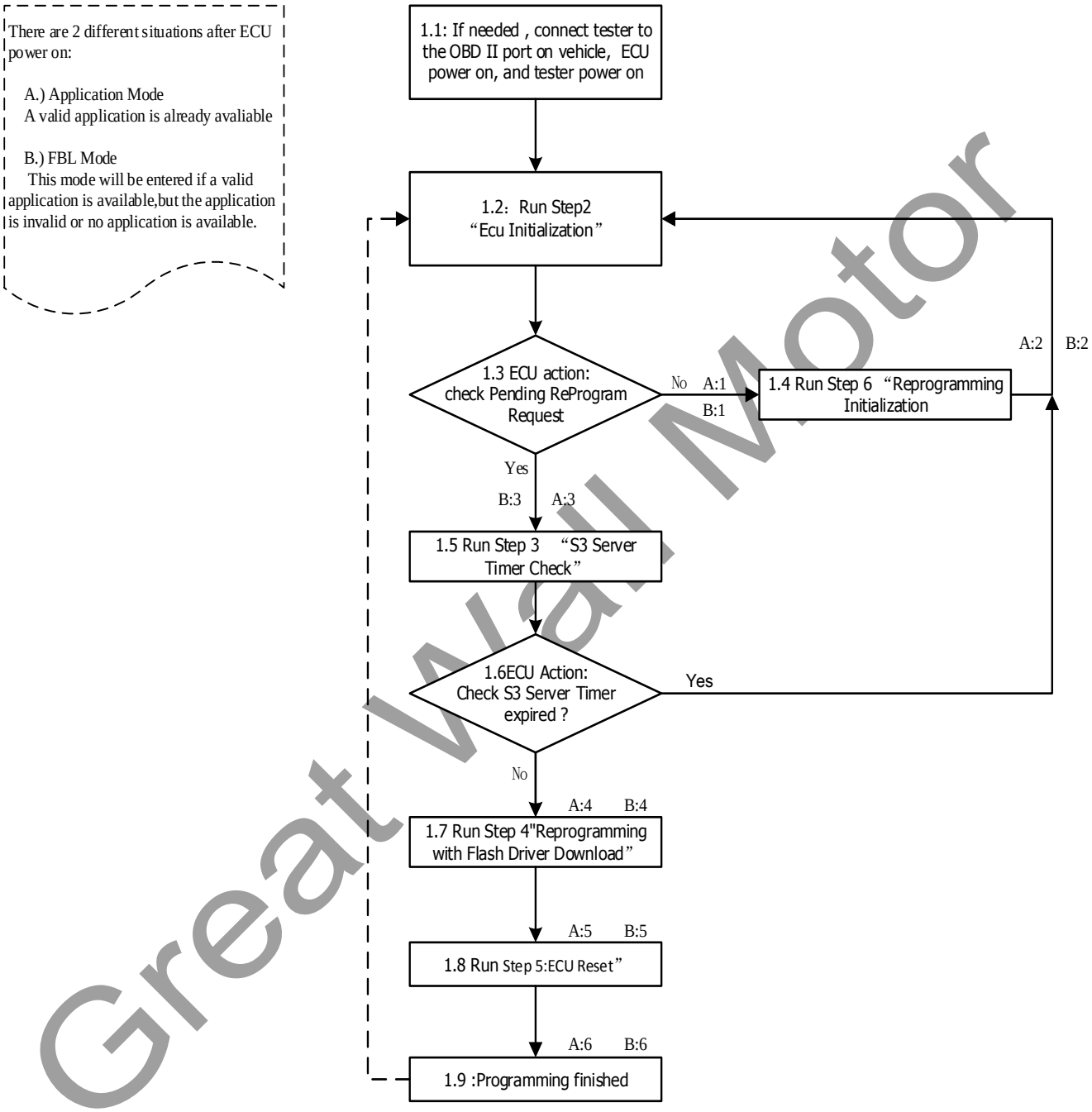

Figure 5: ECU Programming Control Flow

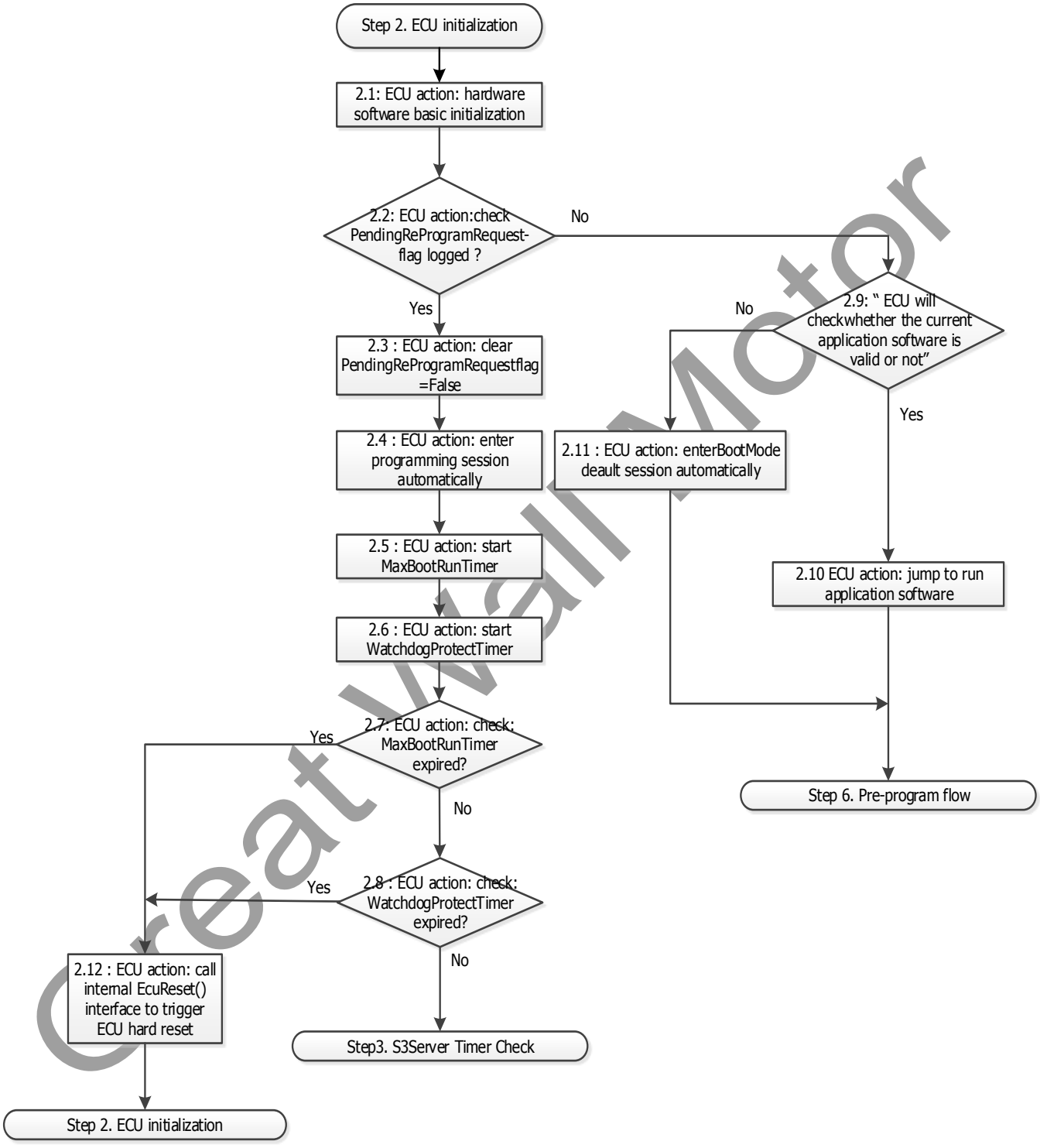

Figure 6: Step 2: ECU Initialization

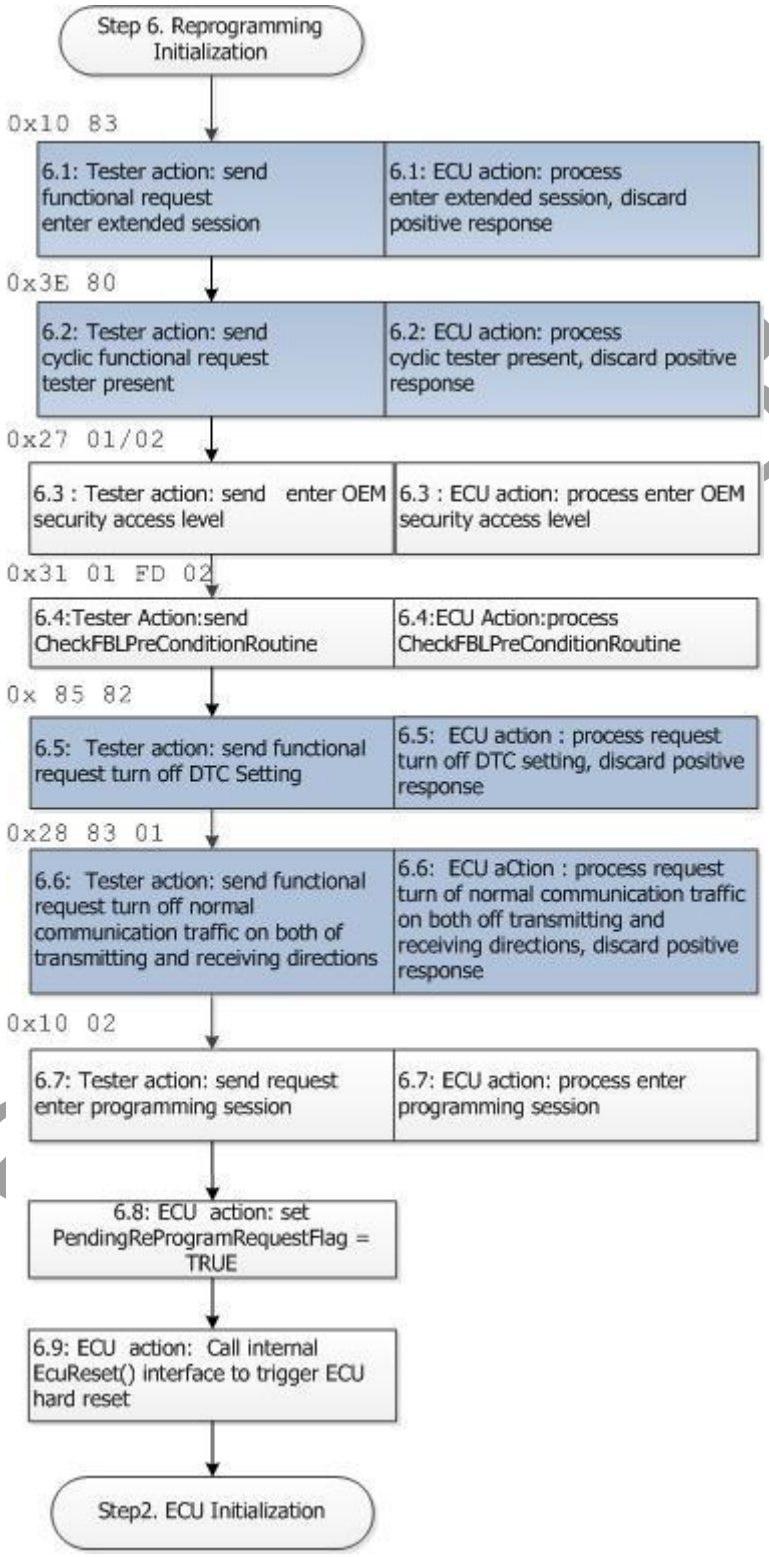

Figure 7: Step 6: Reprogramming Initialization

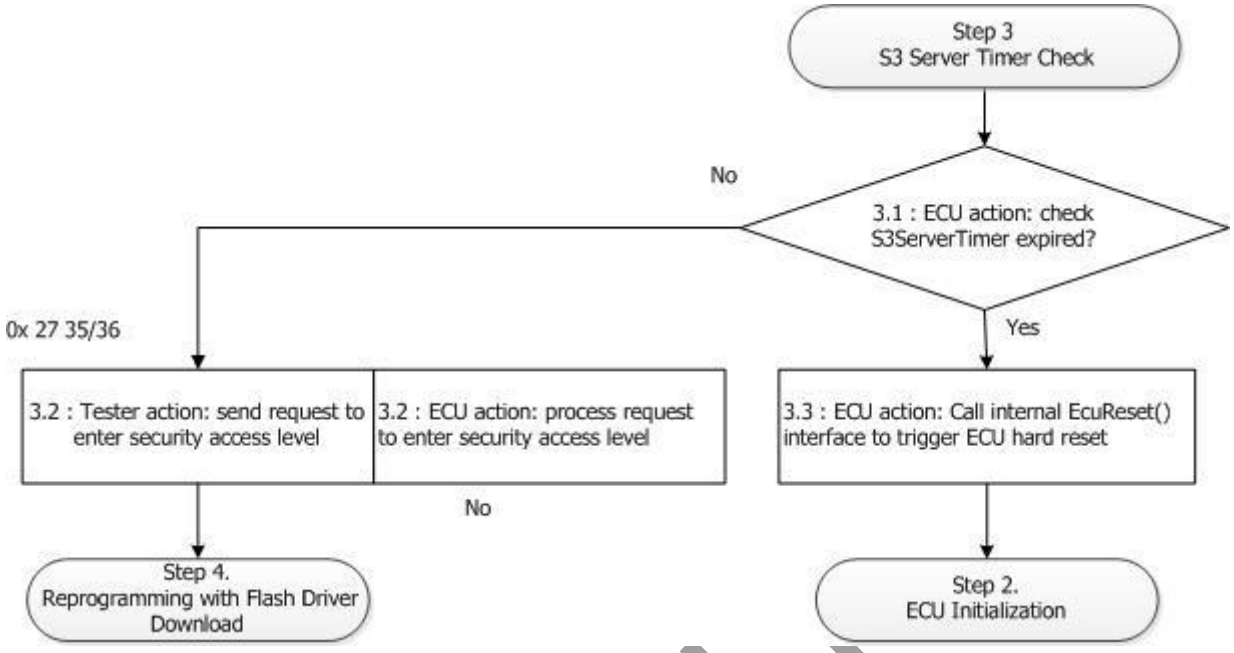

Figure 8: Step 3: S3Server Timer Check

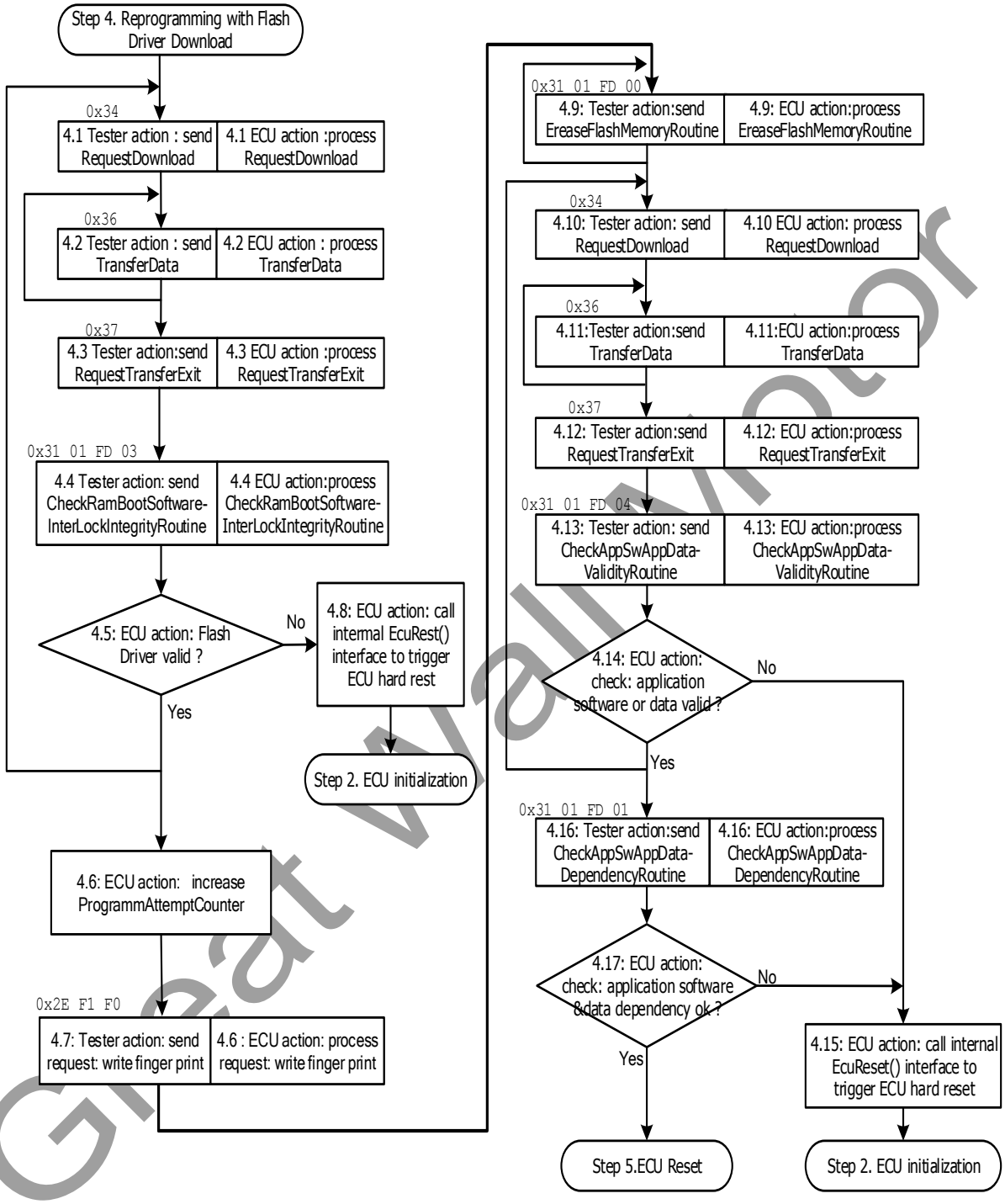

Figure 9: Step 4: Reprogramming with Flash Driver Download

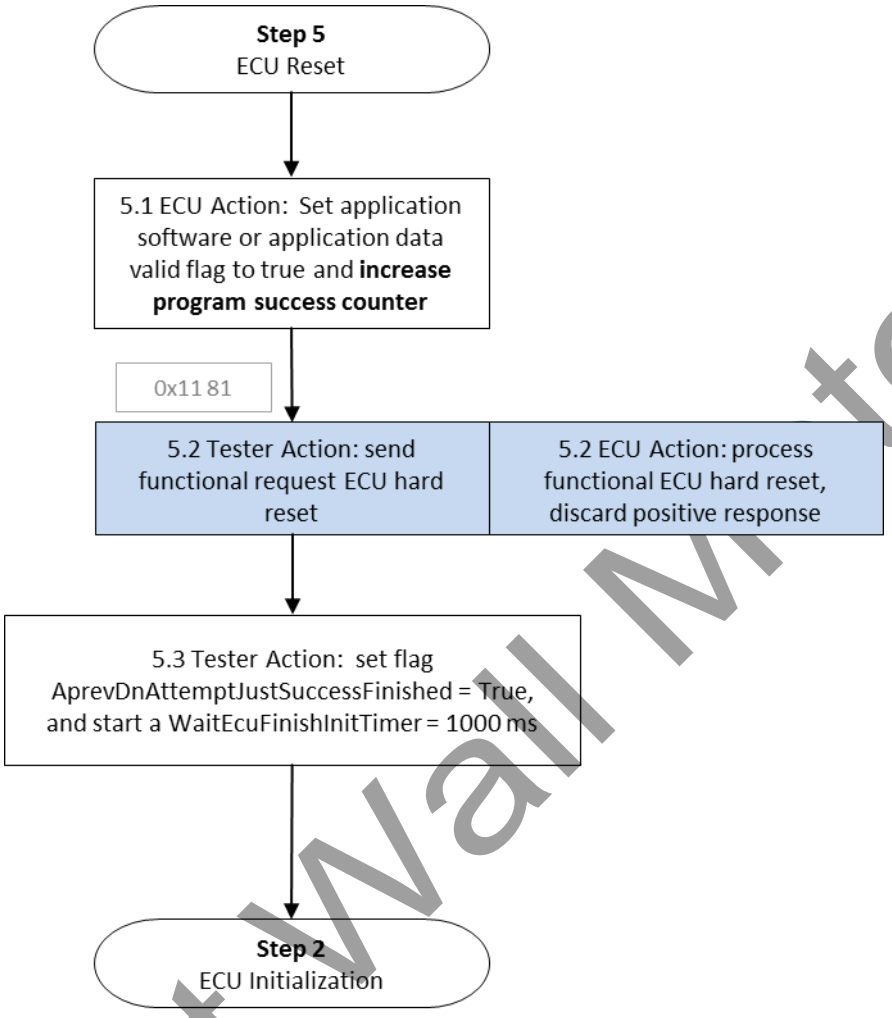

Figure 10: Step 5: ECU Reset

4.4.1 ECU Programming Control flow

RS-FBL-4010 Overview of ECU Programming Control Flow

The overview of the FBL programming procedures is given, to help all of the stakeholders get the complete overview about how FBL works, based on the typical use cases for the flash bootloader.

The whole procedure will start from the sub-step 1.1. If needed, we always assume the procedure is started from connecting the tester to the OBD II port on vehicle, and make sure the ECU and tester both power on. Then in the next sub-step 1.2 the procedure will run into Step 2 “ECU Initialization”.

After ECU finished the initialization steps then it will run boot software (see Mode B) or application software (see Mode A) respectively. All these actions are executed automatically, without any tester interaction, and then ECU waits for the testers UDS communication.

If the tester starts the programming sequence, Step 6 “Reprogramming Initialization” will be executed. In Step 6 the ECU will be prepared for the reprogramming sequence (e.g. Communication Control, switch to programming session …). Then the procedure will go to sub-step 1.2, it will run Step 2 “ECU Initialization” again. After these steps the ECU will run in the boot mode for both modes.

In sub-step 1.5 Step 3 “S3Server Timer Check” will be executed. This step will perform the security access within the programming session and starts the programming. Sub-Step 1.7 will start Step 4 “Reprogramming with Flash Driver Download”. First, the flash driver will be downloaded afterwards the data itself. If step 4 was successfully, sub-step1.8 will start step 5 “ECU reset” which performs an ECU hard reset and finishes the programming sequence (boot software post program steps to set the application software valid flag to true, increase the program success counter).

4.4.2 ECU Initialization

RS-FBL-4020 Overview of Programming Step 2 “ECU Initialization”

In the step 2, first the ECU will always run ROM version FBL software image at each time of hard reset. In the sub-step 2.1 ECU will do the basic hardware and software initialization which are necessary. Then the procedure will go to sub-step 2.2, ECU ROM version FBL software will check if a previous pendingReProgramRequestFlag is logged or not, in the flash or EEPROM. If yes, a reprogramming request has already been issued by the tester in the application mode, then ECU will keep running in the boot mode, and will go to the sub-step2.3. If no pendingReProgramRequestFlag is logged, then the procedure will go to the sub-step 2.9. The ECU will check whether the current application software is valid or not. If yes, there is already valid application software existing in ECU, then ECU will go to sub-step 2.10 to run the application software. After the sub-step 2.10, the ECU already run the application software and wait for the tester to send an UDS requests. If no valid application software exists in ECU, then ECU will go back to boot mode immediately, and run sub-step 2.11. In the step 2.3, ECU will clear PendingReProgramRequestflag to false and run sub-step 2.4.In the sub-step 2.4, ECU will enter programming session automatically to ready for the following

steps for application software programming or application data programming. Then ECU will go to sub-step 2.5 to start a monitoring timer “MaxBootRunTimer” to control the overall programming time. The timeout value of MaxBootRunTimer is statically configurable by using the macro constant in C code header file, shall be set to 3 times of the normal total programming time. In the next sub-step 2.6, we will use “micro monitoring method” now. ECU will start the WatchdogProtectTimer. This timer is much shorter than the previous timer with the timeout value have only for example 100ms, to make sure the FBL software which is running on ECU will not deadlock. This timer should also be statically configurable by using the macro constant in the C code header file. And the timer is also checked and served in the cyclic task, just with much faster period, for example, 30ms cyclic task. In the following sub-step 2.7, ECU will check if this MaxBootRunTimer is expired. If yes, then the whole programming procedure have the very critical issue, we must exit this programming attempt and trigger a internal ECU hard reset in the following sub-step 2.12. In the next sub-step 2.8, ECU will check if the WatchdogProtectTimer is expired or not periodically. If yes, then it will trigger a hard reset unconditionally in the sub-step 2.12 to make sure the FBL software will not be dead-locked.
Note: The MaxBootRunTimer should be defined according the worst case estimates for aging of the flash memory including a proper buffer.

RS-FBL-4030 System Initialization

For the system initialization requirement, the below checkingList table is applied

No	Action List	Description	Rationale
2.1	1).CAN initialization	To initialize the CAN communication interface or port used for UDS and FBL downloading.	Mandatory, could be simplified than the full-feature CAN driver initialization in FBL.
	2). LIN initialization	To initialize the LIN communication interface and its driver used for UDS and FBL downloading.	Mandatory, could be simplified than the full-feature LIN driver initialization in FBL.
	3).Flash initialization	To initialize the code flash as well as data flash (if available and is used by FBL).	Mandatory.
	4).Watchdog initialization	To init the watchdog designed in the hardware device.	Mandatory.
	5).The EEPROM or emulated EEPROM initialization	To intiate the EEPROM module.	Mandatory.
	6).MCU itself initialization	Included, but not limited : 1). Register initialization;	Mandatory, if the respective devices is

		2). The interrupt vector initialization (if the interrupt is used in FBL); 3). PLL clock initialization; 4). The peripherials initialization (if is used in FBL)	used in the FBL.
	7).Operating System Initialization	To initialize the operating system (if is used in FBL).	If the operating system is used, this step is mandatory; otherwise optional.
	8).UDS Protocol Stack Initialization	To initialize the UDS protocol stack.	Mandatory
	9).FBL Manager Initialization	To initialize the downloading procedure state manager module	Mandatory

RS-FBL-4040 Requirement PendingReProgramRequestFlag

For the system requirement related to the PendingReProgramRequestFlag, the below checkingList table is applied

SubStep No	Action List	Description	Rationale
2.2	ECU action: check PendingReProgram-RequestFlag logged ?	If the PendingReProgram-RequestFlag is stored in the flash or EEPROM, should be decided by the exact design of the FBL in ECU side. From the point of view of system requirement, it requires only this PendingReProgram- RequestFlag should be a NVRAM variable, i.e. its value can be survive between the power on reset or hard reset. The length and content of this flag could also be decided by the ECU actual design. For Example, this flag can be defined as 8 bytes length, for all of 0x00 and all of 0xFF, this flag means FALSE, i.e. no reprogramming request is logged. If the string of "FBLdnreq" ASCII value are set for this flag, means a re-programming request has been issued by tester. After the boot software running mode is entered, then this flag should be cleared	Mandatory,must be stored in the NVRAM, the content can be survived between hard reset.

		once time, means, this flag must be consumed to make sure the boot mode will only be entered once time for each time of re-programming request.	

RS-FBL-4050 Requirement WatchDogProtectTimer

For the system requirement related to the MaxBootRunTimer and WatchdogProtectTimer, the below checkingList table is applied.

SubStep No	Action List	Description	Rationale
2.5	ECU action: start MaxBootRunTimer	This timer is a "Macro monitoring method" to make sure the whole of system, including tester, the programmed ECU, and the other related ECU, such as gateway ECUs, should finish the whole of the programming procedure in the reasonable time. The value of the timer is defined as 3 times of the normal total programming time for the programmed ECU, default value is 30 minutes, and could be "calibrated" in the actual programming procedure.	Mandatory, to make sure the whole of the programming procedure will not be blocked due to the "macro reason"
2.6	ECU action : start WatchdogProtect-Timer	This timer is a "Micro monitoring method" to make sure the ECU boot software will not be dead-locked due to software reason. This timer is much shorter than the previous timer, and the default value is 100 ms. This timer should be designed and implemented in the hardware solution, and it should be checked and served in the 30ms cyclic task.	Mandatory, to make sure the ECU boot software will not be blocked due to the "micro reason".

RS-FBL-4060 Requirement ApplicastionSoftwareValidFlag

For the system requirement related to the ApplicastionSoftwareValidFlag, the below checkingList table is applied.

SubStep No	Action List	Description	Rationale

2.9	ECU action: check ApplicationSoftware ValidFlag?	If the ApplicatoinSoftwareValid-Flag is stored in the flash or EEPROM , should be decided by the exact design of the FBL in ECU side. From the point of view of system requirement, it requires only this ApplicationSoftwareValid Flag should be a NVRAM variable, i.e. its value can be survived between the power on reset or hard reset. The length and content of this flag could also be decided by the ECU actual design. For Example, this flag can be defined as 8 bytes length, for all of 0x00 and all of 0xFF, this flag means FALSE, i.e. no valid application software exists. If the string of "Aswdatok" ASCII value are set for this flag, means a valid application software has already been existed on this ECU. Each time of hard reset, the boot software will be run first, and this flag will be checked by bootsoftware, to decide if the jumping to application software should be made. This flag should be kept as valid value pattern, until the next time of programming attempt, this flag will be cleared as all 0x00 or all 0xFF.	Mandatory, must be stored in the NVRAM, the content can be survived between hard reset.

4.4.3 S3Server Timer Check

RS-FBL-4070 Overview of Programming Step 3 "S3Server Timer Check"

In the step 3, the pre-program steps in the boot mode will be executed. Due to the programming session mode has been entered automatically in the sub-step 2.4, we shall make sure the programming session not exit automatically due to the S3Server timer expired at ECU side in the sub-step 3.1. And the time out will be checked in the fast UDS cyclic task such as 10ms cyclic task. This time out checking will be executed periodically in the whole programming procedure after the programming session has been entered until the programming procedure has been finished. After finished, the ECU hard reset will be executed again. So the time out checking doesn't only run in the sub-step 3.1, but also for all of the programming steps. Each time when ECU receives any request from tester side, and there must be some very critical problem has been happened in the UDS protocol stack, so ECU will go to the sub-step 3.3 to call an internal ECU Reset interface to trigger an ECU hard reset. ECU will go back to the Step 2 "ECU Initialization".

In most of the normal UDS communication situation, this S3Server timer will not expire. So the programming procedure will go to the sub-step 3.2. The ECU is already initialized for the reprogramming sequence the tester will send 0x27 security access service to enter the boot security access and Step 4 "Reprogramming with Flash Driver Download" will be entered. If passed, this means this tester is a valid tester which is allowed to execute the following main boot software program steps described in the step 4.

RS-FBL-4080 Security Access

The below table describe the sub-step 3.2 related information.

Service and Sub-function	Message format	Description
0x27 0x35	Request message: Total request length 2 bytes. 0x27 35	request seeds for Boot Security Access.
0x67 0x35	Response message; Total response length 6 bytes. 0x67 35 SeedByte1 SeedByte2 SeedByte3 SeedByte4	The response from ECU for the request seeds, the 4 bytes random-generated seeds will be sent from ECU to tester for use.
0x27 0x36	Request message , total request length is 6 bytes: 27 36 KeyByte1 KeyByte2 KeyByte3 KeyByte4	The request to submit key. The key will be produced by using the encryption algorithm. The security access algorithm Refer to REF[10] . The Security mask of security algorithm Refer to ECU Level Diagnostic Requirement.
0x67 0x36	Response message , total response length is 2 bytes: 0x67 36	If the submitted keys are matched exactly with the keys calculated in ECU locally, then this means this tester is a valid tester, then a positive response will be returned to tester side.

4.4.4 Reprogramming with Flash Driver Download

RS-FBL-4090 Overview of Programming Step 4 "Reprogramming with Flash Driver Download"

In the step 4, the main program steps in the boot software will be executed.

First the RAM version Flash Driver will be downloaded from tester side into ECU RAM, then jump to it to run. We shall support the non-continuous RAM segments downloading to make sure the Flash Driver can be designed as located in the different RAM segments in the ECU. The downloading is executed based on each segments one by one. For each segments, the downloading begin from the sub-step 4.1. Tester will first send the 0x34 request to execute Request Download to ask ECU to do preparing action and allocate the resource to receive the following downloaded data, then in the sub-step 4.2 the 0x36 TransferData service will be used to transfer downloaded data multi-times, until the transferring for this segment is finished. Then in the sub-step 4.3 the 0x37 RequestTransferExit service will be used to exit of the request download.Then the tester will go the sub-step 4.4 to send the 0x31 RoutineControl service to run checkFlashDriverValidityRoutine. If there are other segments will be downloaded, then the sub-step from 4.1 to 4.4 will be repeated, until all of the segments downloading are finished.The detailed actions on how to check the FlashDriverValidity will be explained in table below. And in the sub-step 4.5, ECU will check if the Flash Driver is valid or not. If not, then this means the downloaded Flash Driver have critical issue and must not be used. So this time of programming attempt is failed, and ECU will go to the step 2 to call the internal ECU reset interface to trigger a ECU hard reset. If the FlashDriverValidity checking is valid, then ECU will send back the checking result in the positive response message to tester side. When the Flash Driver runs, the ECU will go to the sub-step 4.6 in which ECU will increase the FBL managing ProgrammAttemptCounter by 1 to record the total number of programming attempt. After this sub-step is executed successfully, the programming procedure will go to the sub-step 4.7 and tester will send the request of 0x2E F1 F0 to write the important finger print data into ECU. After this request is received by ECU, ECU needs to judge the ProgrammingAttemptCounter value and MaxAllowableProgrammingTimes; If the ProgrammingAttemptCounter value is smaller than or equal to MaxAllowableProgrammingTimes, a positive response will be sent and the ECU must write the finger print into the EEPROM and go to the next sub-step 4.9; If the ProgrammingAttemptCounter value is larger than MaxAllowableProgrammingTimes, the NRC will be sent and the reprograming process will be ceased.

In this step, tester will send the request of EraseFlashMemoryRoutine for each of the independent flash segments or sectors one by one to make sure the flash segments or sectors will be the default "initial" value, i.e. all of 0x00u or 0xFFu. The request format is 0x31 01 FD 00 optional parameters. If the erase memory routine is received by ECU, the application valid flag is set to be invalid by the ECU. The detailed exact request format will be described in the below table. After each flash segments or sectors are erased successfully, the procedure will go to the next group of the sub-steps. For each of independent flash sectors, the tester will go the sub-step 4.10 to send the request of RequestDownload to ECU. After ECU receives , it should do all of the needed preparing actions like allocate the resources, and return the acceptable size of the segment from ECU side to the tester side in the positive response message. Then tester will go to the sub-step 4.11 to send the request of

TransferData multi-times until this flash segment is transferred successfully, then the tester will go to the sub-step 4.12 to send the request of RequestTransferExit for this flash segment downloading. Then the tester will go to the next sub-step 4.13 to send the request CheckAppSwApp-DataValidityRoutine to verify the correctness of this flash segment which is just downloaded from tester side into the flash of the target ECU flash device. The detailed parameters and format for this routine will be described in the table below. Then the ECU will go to the next sub-step 4.14 to check whether application software or application data is valid or not. If invalid, then ECU will go to the next sub-step 4.15 to call an internal ECU Reset interface to trigger an ECU hard reset. After reset is executed, the procedure will go to Step 2 ECU initialization. If this validity check is passed, the ECU will return the positive response to tester side and then tester will check if there are still other flash blocks which are waiting for downloading. If yes, it will repeat the above sub-steps from 4.10 to 4.14 until all of the flash blocks downloading are finished successfully one by one.

After all of the necessary flash blocks are downloaded successfully, the procedure will go to the sub-step 4.16. The tester will send the request of CheckAppSwAppDataDependencyRoutine to request ECU do the dependency checking between the application software and application data. If the application software and application data are combined into a single binary image and downloaded at once, no need to do this step. If the application software or application data are downloaded independently, then we should check if they are dependent on each other. If yes, this checking routine will be passed and the whole programming procedure will go to Step 5 ECU Reset Step. If not, this means the downloaded application software or application data can't match and cooperate with each other perfectly, so we must exit this time of programming attempt. So the procedure will go to the sub-step 4.15 , ECU will call internal ECU reset interface to trigger a time of ECU hard reset, after which the step 2 ECU initialization after hard reset will be executed again.

Note: a segment refers to a continuous flash address range in a flash file;

RS-FBL-4100 Download Services

For the system requirement related to the sub-steps 4.1, 4.2, 4.3 the below checkingList table is applied.

UDS Service	Sending Channel	The Request	The Response	Description
0x34 Request-Down-load 0x34	Physical Addressing Channel	The whole request length is 11 bytes. 0x34 00 44 AddByte1, AddByte2, AddByte3, AddByte4, SizeByte1, SizeByte2, SizeByte3, SizeByte4	The whole response length is 6 bytes. 0x74 40 MaxNumOfBlockLenByte1 MaxNumOfBlockLenByte2 MaxNumOfBlockLenByte3 MaxNumOfBlockLenByte4 The Block Length have 4 bytes, Byte order is U32_BlockLen = (unsigned int) (BlockLenByte1<< 24u |	At the tester side and the ECU side, the below byte order is used U32_Addr = (unsigned int) (AddByte1<< 24u | AddByte2 << 16u | AddByte3 << 8u |

			BlockLenByte2 << 16u | BlockLenByte3 << 8u | BlockLenByte4);	AddByte4); U32_Size = (unsigned int) (SizeByte1<< 24u | SizeByte2 << 16u | SizeByte3 << 8u | SizeByte4);
0x36 Transfer-Data	Physical Addressing Channel	The whole request length is 2 + actual length bytes or 2 + BlockLen. 0x36 SN DataByte1 DataByte2 DataByte3 …. SN : is a value between 0x00-0xFF, for 1st round , should be begun at 1, and next round begun at 0; DataByte1 DataByte2… are the actual transferred data.	The whole of response length is 2 bytes 0x76 SN The SN number must be match with the SN in the request.	If the actual length of data is less than allocated BlockLen, or just a last block , then the total request length is 2 + actual length; otherwise, the length should be 2 +BlockLen, to increase the performance of traffic.
0x37 Request-Transfer-Exit	Physical Addressing Channel	The whole request length is 1 byte.	The whole of response length is 1 byte : 0x77.	

RS-FBL-4110 Routine CheckFlashDrivervalidityRoutine

For the system requirement related to the sub-steps 4.4Run CheckFlashDrivervalidityRoutine, the below checkingList table is applied.

UDS Service	Sending Channel	The Request	The Response	Description
0x31 Routine-Control	Physical Addressing	The whole request length is 8 bytes.	The whole response length	The functionalities for this routine have two

0x31	Channel.	0x31 01 FD 03 CRC32Check-Sum, 4 bytes length.	is 5 bytes.	main parts To calculate the CRC32 checksum at ECU side too, and compare the calculated CRC32 checksum with the received CRC32 check sum, must match. The CRC32 algorithm, please refer to the chapter 6.

RS-FBL-4120 Routine ProgrammAttemptCounter

For the system requirement related to the sub-steps 4.6 ProgrammAttemptCounter, the below checkingList table is applied.

The DataStructure	Description
ProgrammAttempt-Counter	32bit unsigned integer, stored in the EEPROM, after End Of Line of ECU production process, it should be initialized to 0x00000000u, for each time of programming attempt, no matter successful or failed, it should be increased by 1, this will be used to trace the total number of the programming attempt.

RS-FBL-4130 Service Write Fingerprint

For the system requirement related to the sub-steps 4.7 write FingerPrint, the below checkingList table is applied.

UDS Service UDS	Sending Channel	The Request	The Response	Description
0x2E WriteData-ByIdentifer	Physical Addressing Channel	The whole request length is 29 bytes. 0x2E F1 F0 Payload Length 26 bytes Structure, from highest byte to lowest byte : Byte0~9: 10 bytes Tester Supplier Id and the tester Serial Number, coded in ASCII ; Byte0~Byte3: 4 bytes: The Tester Supplier id. Byte4~Byte9: 6 bytes: the SN number of tester Byte10~Byte15: 6 bytes:	The whole response length is 3 bytes.	The storing location for the finger print should be in EEPROM, and all of the information should be prepared by tester when the programming attempt is executed.

		the flashing operation date and time, coded in BCD. Byte 10: year offset (current year – 2000). Byte11: month. Byte12: day. Byte13: Hour, 24hours format. Byte14: Minute. Byte15: Seconds. Byte16~19: 4 bytes, the ID of 4S shop or garage coding, for tracing the location of flashing, coded in ASCII. Byte 20~25: 6 bytes : Reserved Data for the OEM or ECU Supplier specific Finger Print data.		

RS-FBL-4140 Routine EraseFlashMemoryRoutine

For the system requirement related to the sub-steps 4.9 Run EraseFlashMemoryRoutine, the below checkingList table is applied.

UDS Service UDS	Sending Channel	The Request	The Response	Description
0x31 Routine-Control 0x31	Physical Addressing Channel.	The whole request length is 13 bytes. 0x31 01 FD 00 44 FlashAddToErase; FlashSizeToErase; 44 means the length of add is 4 bytes, and the length of size is also 4 bytes. 0x44 1). Flash AddToErase: 4 bytes length, unsigned 32bits integer coding , from highest byte to lowest byte: 2). FlashSizeTo-Erase, 4 bytes length, unsigned 32 bits integer codding, from highest byte to lowest byte.	The whole response length is 5 bytes. 0x71 01 FD 00 00erase successful 0x71 01 FD 00 01 erase failure.	Tester use this request to erase the flash sectors one by one, the validation check for FlashAddToErase, and FlashSizeToErase, should be checked at ECU side.

RS-FBL-4150 Routine CheckAppSwAppDataValidity

For the system requirements related to the sub-steps 4.10 to 4.12, which are similar as the respective sub-steps from 4.1 to 4.3, so that checking List table is also applied.

For the system requirement related to the sub-steps 4.13 Run CheckAppSwAppData-ValidityRoutine, the below checkingList table is applied.

UDS Service UDS	Sending Channel	The Request	The Response	Description
0x31 Routine-Control 0x31	Physical Addressing Channel	The whole request length is 17 bytes. 0x31 01 FD 04 44 FlashAddToCheck FlashSizeToCheck CRC32CheckSum 44 means the length of add is 4 bytes, and the length of size is also 4 bytes. 1). Flash AddToCheck: 4 bytes length, unsigned 32bits integer coding, from highest byte to lowest byte. 2). FlashSizeTo-Check, 4 bytes length, unsigned 32 bits integer coding, from highest byte to lowest byte. 3). CRC32Check-Sum, 4 bytes length,	The whole response length is 5 bytes. 0x71 01 FD 04 Result	Tester should calculate the CRC32 checksum in the given range, put in the last of the request, then send to ECU side. And ECU should also calculate the CRC32CheckSum, then compare the received with the calculated value, if match, then validation check is passed, otherwise it is failed. The CRC32 algorithm , please refer to the chapter 6 .

RS-FBL-4160 Routine CheckAppSwAppDataDependency

For the system requirement related to the sub-steps 4.16 Run CheckAppSwAppData-DependencyRoutine, the below checkingList table is applied.

UDS Service UDS	Sending Channel	The Request	The Response	Description
0x31 Routine-Control 0x31	Physical Addressing Channel	The whole request length is 4 bytes. 0x31 01 FD 01	The whole response length is 5 bytes. 0x71 01 FD 01 Result	If the application software and application data are built in a single binary image file, and downloaded once time, then this routine should return OK every

				time. If the application software or application data is downloaded independently, then their software id and data id , and their software version and data version , should be checked, and must be matched with each other. These data is stored in the fixed offset address for the application software or application data.

4.4.5 ECU Reset

RS-FBL-4170 Overview of Programming Step 5 "ECU Reset"

In the step 5, the post program steps in the boot software will be executed. At the sub-step 5.1, ECU will set the ApplicationSoftwareValidFlag or ApplicationDataValidFlag to TRUE and increase the programSuccessCounter by 1. After this sub-step, tester will go to the sub-step 5.2 to send the request of 0x11 81 on the functional addressing channel to request all of the ECU do hard reset action. Then after all of the ECU, including the ECU which is under programmed, receives this functional addressing request, they will do the hard reset action, and discard the positive response. Then in the next sub-step 5.3, tester will set an intenal flag AprevDnAttemptJustSuccessFinished. Then tester will start a WaitEcuFinishInitTimer to wait the ECU to finish their initialization action and run the application software successfully. Then the whole procedure will go to the Step 2 "ECU Initialization".

RS-FBL-4180 Requirement ProgrammSuccessCounter

For the system requirement related to the sub-steps 5.1 ProgrammSuccessCounter, the below checkingList table is applied.

The DataStructure	Description
ProgrammSuccess-Counter	32bit unsigned integer, stored in the EEPROM, after End Of Line of ECU production process, it should be initialized to 0x00000000u. For each time of programming attempt which has been finished successfully, it should be increased by 1, this will be used to trace the total number of the successful programming attempt.

RS-FBL-4190 Service EcuReset

For the system requirement related to the sub-steps 5.2 run ECU hard reset on the functional

addressing channel, the below checkingList table is applied.

UDS Service UDS	Sending Channel	The Request	The Response	Description
0x11 EcuReset 0x11	Functional Addressing Channel	The whole request length is 2 bytes. 0x11 81	All of the ECUs process the received request, but discard the positive response.	All of the ECUs, including non-programmed and the programmed currently, should execute the hard reset action, and after ECU initialization after hard reset, all of ECUs will enter the default session automatically, restore the DTC setting, and the bi-directional common interaction layer traffic, which has been turned off in the programming procedure.

RS-FBL-4200 Requirement AprevDnAttemptJust-SuccessFinished

For the system requirement related to the sub-steps 5.3 AprevDnAttemptJust-SuccessFinished and WaitEcuFinishInitTimer, the below checkingList table is applied.

The DataStructure	Description
AprevDnAttemptJustSuccesFinished	8bit unsigned Boolean, stored in the RAM, it will be set to TRUE, after the each time of the programming attempt is just finished successfully. Then after the ECU initialization after hard reset is finished, and application software is running, this flag will be consumed, i.e. clear to FALSE, before to execute one time of clear DTC request from tester.
WaitEcuFinishInitTimer	32bit unsigned integer, stored in the RAM, it will be set to 1000ms, after the request of 0x11 81 is sent out. It can be ticked in the 100ms cyclic task on tester side. And after this timer is expired, we assume all of the ECUs have finished their initialization, then we can run the next UDS services.

4.4.6 Reprogramming Initialization

RS-FBL-4210 Overview of Programming Step 6 “Reprogramming Initialization”

In the Step 6 the tester will prepare the ECU for a new version of the application software or

application data.
The procedure will go to sub step 6.1, in which tester will send the request of 0x10 83 on the functional addressing channel to all of the ECUs to request all of the ECUs, including the non-programmed ECU and the programmed ECU, to enter the 0x03 extended session in the same time. After ECUs receive this request, then they must enter the extended session mode and discard the positive response. Then the procedure will go to sub step 6.2, and tester will send the request 0x3E 80 cyclically on the functional addressing channel to make sure all ECUs kept in the 0x03 extended session. After ECUs receive this request, they will always reload its S3Server Timer and then discard the positive response. These cyclic requests of tester present must run continuously for the whole programming procedure until all programming steps in the application mode and boot mode are finished successfully. The period for sending this 0x3E 80 request should be 2000ms by default. Afterwards the procedure will go to sub-step 6.3 to perform the security access for levels 0x01/0x02. Afterwards the procedure will go to the sub-step 6.4 in which the tester will send the request 0x31 01 FD 02 to the programmed ECU to run CheckFBLPreConditionRoutine. After ECU receives this request, it will run this CheckFBLPreConditionRoutine and if all of the pre-conditions are passed, ECU must send a positive response message to tester.
The tester will go to the sub-step 6.5 after receives this positive response. It will then send the request 0x85 82 on the functional addressing channel to all of the ECU to turn off the DTC setting globally. All ECUs who receive this request must turn off the DTC setting, i.e. not log the DTC any more, and discard the positive response message. Then tester will go to the sub-step 6.6 to send request 0x28 83 01 on the functional addressing channel to all of the ECU to turn off normal interaction layer communication traffic on both transmitting and receiving directions to make sure most of the network bandwidth can be reserved for the following programming traffic. ECU must discard the positive response. Then all of the necessary preparing actions in the application mode have been finished before switching back into the boot mode.
Then the tester will go to sub-step 6.7 to send request 0x10 02 on the physical addressing channel to the programmed ECU. The ECU must enter the 0x02 programming session and send the positive response to tester side. Then in next sub-step 6.8, ECU will set the flag PendingReProgram-RequestFlag to TRUE and then go to sub-step 6.9 to call an internal ECUReset interface to trigger a hard reset. Then the procedure will go to step 2 “ECU Initialization”. ECU will enter the boot mode after initialization which means the switching from the application mode into boot mode is finished.

RS-FBL-4220 Service DiagnosticSessionControl
For the system requirement related to the sub-steps 6.1 run Request to enter extended session on the functional addressing channel, the below checking List table is applied.

UDS Service UDS	Sending Channel	The Request	The Response	Description
0x10 Diagnostic-Session-Control	Functional Addressing Channel	The whole request length is 2 bytes. 0x10 83	All of the ECUs process the received request, but discard the positive response.	Basic requirement for all of ECUs.

0x10				

RS-FBL-4230 Service TesterPresent

For the system requirement related to the sub-steps 6.2 run Request tester present on the functional addressing channel, the below checking List table is applied.

UDS Service UDS	Sending Channel	The Request	The Response	Description
0x3E Tester-Present 0x3E	Functional Addressing Channel	The whole request length is 2 bytes. 0x3E 80	All of the ECUs process the received request, but discard the positive response.	Basic requirement for all of ECUs. By default, This request should be sent cyclically at periodic 2000ms timer base. When the ECU receive this request or any other request, it should reload its S3ServerTimer to 5000ms by default.

RS-FBL-4240 Service SecurityAccess

For the system requirement related to the sub-steps 6.3 run Request to enter OEM security access level on the physical addressing channel, the below checking list table is applied.

Service and Sub-function	Message format	Description
0x27 0x01	Request message: Total request length 2 bytes. 0x27 01	request seeds for OEM Security Access Level 2.
0x67 0x01	Response message; Total response length 6 bytes. 0x67 01 SeedByte1 SeedByte2 SeedByte3 SeedByte4	The response from ECU for the request seeds, the 4 bytes random-generated seeds will be sent from ECU to tester for use.
0x27 0x02	Request message , total request length is 6 bytes: 27 02 KeyByte1 KeyByte2 KeyByte3 KeyByte4	The request to submit key. The key will be produced by using the encryption algorithm. The security access algorithm Refer to REF [10]. The Security mask of security algorithm Refer to ECU Level Diagnostic Requirement.
0x67 0x02	Response message , total response length is 2 bytes: 0x67 02	If the submitted keys are matched exactly with the keys calculated in ECU locally, then this means this tester is a valid tester, then a positive response will be returned to tester

		side.

RS-FBL-4250 Check CheckFBLPreCondition

For the system requirement related to the sub-steps 6.4 run 0x31 CheckFBLPreCondition-Routine on the physical addressing channel, the below checking List table is applied.

UDS Service	Sending Channel	The Request	The Response	Description
0x31 Routine-Control 0x31	Physical Addressing Channel	The whole request length is 4 bytes. 0x31 01 FD 02	The whole response length is 5 bytes. 0x71 01 FD 02 Result	For the routine of precondition checking, no optional parameters are defined. The exact pre-conditions should be decided by each ECU specifically, suggest to check the below information. 1). Ignition switch must be located at IgnitionOn, but not cranking 2). SystemBattery-Voltage must be in the valid range : 9-16 Voltage 3). Vehicle speed must be 0kph 4). Engine rotate speed must be 0 rpm 5). ECU must be in the high power mode; 6). ECU must ensure all of the inputs and outputs stay in the safe and secure inactive status.

RS-FBL-4260 Service ControlDTCSetting

For the system requirement related to the sub-steps 6.5 run Request turn off DTC setting on the functional addressing channel, the below checking List table is applied.

UDS Service	Sending Channel	The Request	The Response	Description
0x85 Control-DTCSetting	Functional Addressing Channel	The whole request length is 2 bytes.	All of the ECUs process the received request, but discard the	This request is used to turn off the DTC setting for all of the non-programmed

		0x85 82	positive response.	ECU temporarily.

RS-FBL-4270 Service CommunicationControl

For the system requirement related to the sub-steps 6.6 run Request turn off the bi-direction of common normal application traffic on the functional addressing channel, the below checking List table is applied.

UDS Service	Sending Channel	The Request	The Response	Description
0x28 Communication-Control	Functional Addressing Channel	The whole request length is 3 bytes. 0x28 83 01	All of the ECUs process the received request, but discard the positive response.	This request is used to turn off the bi-direction common normal application traffic for all of the non-programmed ECU temporarily.

5 FBL Requirements

The whole of vehicle bus are consist of CAN network and LIN network. Each network segments have many ECUs, due to the development progression schedule limitation or resources restriction, some ECUs cannot support reprogramming (temporarily), called non-programmable ECU, other ECUs can support, called programmable ECU.

5.1 System Requirements for reprogrammable ECUs

RS-FBL-5010 The robust point: can be restart again if failed

All of programmable ECU, the flash boot loader must be robust enough to start the next time of programming attempt again , if the previous programming is not finished successfully , due to any issue, such as : programming failure of flash erasing or programming , or interrupted abnormally , or a programming timeout , or a reset is happened.

RS-FBL-5020 The boot software can only switch to application software running mode when the downloaded application is valid

All of programmable ECU, the flash boot loader must ensure the software controlling switching from boot software mode to application mode will only be happened there is a valid application software image in the code flash in ECU. If the application software image is missing, or invalid or corrupted due to partially erasing or programming, then FBL must stay in boot mode, and allow downloading application software again in anytime.

RS-FBL-5030 The diagnostic traffic can only be allowed when booting mode, normal traffic should be closed

All of programmable ECU, when running in the boot mode, shall not be impacted by the normal communication of the other ECUs on the vehicle network. In general, no normal communication CAN messages shall be supported in the boot mode.

5.2 ECUs System Requirements for FBL Implementation

RS-FBL-5040 binary image file format requirement

The client tester side flashing tool must support the below binary image file format for the ram boot software, application software, and the application data, The mainstream *.hex, *.s19, *.bin must be supported mandatorily.

RS-FBL-5050 The robustness and reliability requirement of FBL

If the downloading and flashing sequence of FBL cannot be finished normally, no matter due to the CAN communication is broken , or the ECU is reset abnormally, or the flash device failure or the checksum or CRC checking failure. After hard reset or power on reset, the whole of the FBL downloading procedure can always be started from the beginning, until the valid application software and or application data has been downloaded and flashed successfully.

RS-FBL-5060 The power mode of the FBL

FBL will always work in the high power mode of the MCU and ECU, should not design and implement the complex hardware and software system design mechanism to support the low power mode in the FBL booting mode. Allowing the ECU to enters the low power mode when the FBL software is running has very big potential safety and security risk, so the low power mode should only be allowed to enter after the applicaiton software is downloaded and flashed into the ECU successfully.

RS-FBL-5070 The size of the download data block

As for CAN protocol, when the data length of the download data is bigger than or equal to 4093 bytes, the data length of the data block(s) must be 4095 bytes. When the data length of the download data is smaller than 4093 bytes, the download data must be transferred in one block.

6 Checksum Algorithm

RS-FBL-6010 Checksum Algorithm

For the downloaded RAM version Boot software or the FLASH version Application Software and application data, an integrity and validity check of the downloaded data sequences shall be performed.
The generator polynomial is:
G(x) = x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1
The briefication of the G(x) is: “04C11DB7”.
The initial value is FFFFFFFFh.

The ECU that follows the AUTOSAR architecture，the CRC module shall routine based on the IEEE-802.3 CRC32 Ethernet Standard as follow：

CRC result width	32bits
Polynomial	04C11DB7h
Initial value	FFFFFFFFh
Input data reflected	Yes
Result data reflected	Yes
XOR value	FFFFFFFFh
Check	CBF43926h
Magic check*：	2144DF1Ch

Revised record:

Version	Author	Date	Revised Description	Document maturity (Draft/ Release)
1.0	GengJunqing	2013-7-01	First issues	Draft
1.1	GengJunqing	2013-9-06	(1).Add " RS-FBL-0146"and table "7" (2).Update Programming step; (3).Update "RS-FBL-1050"," RS-FBL-1064" (4).change padding byte "0x00"to "0xFF" of "RS-FBL-0456"	Draft
1.2	Liu Bin	2014-2-10	(1).Change the definition of "LIN TP layer time parameters" and "LIN Diagnostic application layer time parameters" from section 3 to section2 (2). Change the title name of section from "Diagnostic Services" to Vehicle Diagnostics Network Requirements (3).Add the description of UDS service in section3 (4).Add "RS-FBL-3010,RS-FBL-3030,RS-FBL-3080,RS-FBL-3090,RS-FBL-3100,RS-FBL-3110,RS-FBL-3120,RS-FBL-3130,RS-FBL-5380,RS-FBL-5390,RS-FBL-5400,RS-FBL-5410,RS-FBL-5420 (5).Update the description of section 4.2 (6).Add the description of 2 and more level flash downloading in section 4.3 (7).Update Programming step; (8).Add the definition of section 4.5 (9).Update the description of RS-FBL-5320 and RS-FBL-5340 (10).Add the definition of section 5.3 (11).Update the requirement number (12).Add the definition of section 6	Release
1.3	Liu Bin	2014-9-2	(1).Change the parameter of "P2*CAN ECU" from 2000ms to 5000ms (2).Change DID of AppSwProgrammingSuccessCntr from F1A4 to F1AA	Release

			(3).Update the ECU Programming Detailed Scenarios Steps (4).Add section 4.6 (5).Add Appendix A and Appendix B	
1.4	Liu Bin	2015-6-10	(1).Add LIN message requirement in section2.3 (2).Divide the content of 2.4 to 2.41 ,2.42 (3).Add LIN bus transport protocol requirement in section 2.42. (4).Add diagnostic requirement of LIN bus in section 2.5. (5).Add the requirement number in section2 (6).Change the requirement number in section3 (7).Change the security level define 27 61/62 to 37 01/02 in table 6 (8).Add DID F1F0 in table8 (9).Change the define of routine DID FD 03 and delete the routine DID FD 05 (10).Delete the requirement of RS-FBL-3060,RS-FBL-3070,RS-FBL-3080, RS-FBL-3090,RS-FBL-3100,RS-FBL-3110, RS-FBL-3120 and RS-FBL-3130 in section3 (11).Delete figure1 in section4.1 (12).Update the define of section4.3 (13).Change the define of programming step 4.4 and 4.5 in section 4.4 (14).Delete the programming step4.17 (15).Delete the programming step 6.5 (16).Delete section4.5 (17).Change the description of “overview of rogramming step2” (18).Change the description of “overview of rogramming step4” (19).Change the description of “overview of rogramming step6” (20).Delete section 5.1 and 5.2 (21).Delete requirement RS-FBL-5410,RS-FBL-5420,RS-FBL-5430	Release
1.5	liuzhandi	2017-4-18	(1) Corrected typo in section 1.1 (2) Renew reference documents in section 1.3 (3)Updated chapter 2: References to latest CAN diagnostic specification v2.8 and LIN	Release

			diagnostic specification 1.1 (4) Updated chapter 3.1 UDS Services (5)Add requirement RS-FBL-3020、 RS-FBL-3030 (6) Updated requirement RS-FBL-3050、 RS-FBL-4240 the 0x27 security access service (7) Updated chapter 3.2 Data Identifiers (8) Updated chapter 3.3 Routines (9) Updated chapter 4.3 (10) Updated chapter 4.4: Figure5、Figure 6、Figure 7、Figure 8、Figure 9 and requirement RS-FBL-4010、RS-FBL-4020、RS-FBL-4070 、RS-FBL-4090、RS-FBL-4210 description accordingly (11)Updated requirement RS-FBL-4080：the security mask description (12) Updated requirement RS-FBL-4100 :0x36 Transfer-Data (13) Updated requirement RS-FBL-4130 extended parameter “testerSerialNumber” to 6 Bytes, shortened parameter “supplierSpecificFingerPrintData” for 2 Bytes (14) Updated requirement RS-FBL-4110、RS-FBL-4150、 RS-FBL-4160、 RS-FBL-4240：Add the response (15) Updated requirement RS-FBL-4250 check information (16)Add requirement RS-FBL-5070“ CAN data block description” and RS-FBL-5080 “ Force Boot Mode description” (17) Updated chapter 6: referencing specification “GWMLAN00-16Security_Algorithm_for_FBL-V1.1” (18) Updated chapter 7 “Checksum Algorithm”	
1.6	Liuzhandi	2018.4.27	(1).Update FBL requirement RS-FBL-3040； (2).Update Figure6 ： Step2 ECU Initialization and RS-FBL-4020 Overview of Programming Step 2 “ECU Initialization”; (3). Delete note in Figure **7:** Step 6: Reprogramming Initialization ;	Release

			(4). Update Figure8 ： Step3 S3 Server Timer Check and RS-FBL-4070 Overview of Programming Step 23“S3 Server Timer Check”; (5). Update Figure9 ： Step4 Reprogramming with Flash Driver Download and RS-FBL-4080 Overview of Programming Step 4 “Reprogramming with Flash Driver Download”; (6). Delete FBL requirement RS-FBL-5080; (7).Update RS-FBL-7010 Checksum Algorithm; (8).Update Service 27 36 in Figure2: Pre-Programming sequence; (9).Update FBL requirement RS-FBL -3040.	
1.7	Zhoujiancang	2019.11.04	(1). In the part of RS-FBL-3040,“ cold ” was changed to“ hard” ; (2).In the part of “RS-FBL-3100 ” the values of yearOffset, month, day, hour, minute, seconds were updated; (3). Update Figure 5: ECU Programming Control Flow; (4). Update Figure 9: Step 4: Reprogramming with Flash Driver Download; (5). In RS-FBL-4090, Update the content; (6). In RS-FBL-4110 and RS-FBL-4150, Change “Appendix1” to “chapter 7”; (7). In the RS-FBL-4130 Service Write Fingerprint, change the Year “1900” to “2000”; (8). Delete the chapter 6.	Release
1.8	Zhoujiancang	2020.10.29	(1). Adding the download process for one flash driver file in multiple segments in “Figure 3: Flash Driver Download” and “Figure 9: Step 4: Reprogramming with Flash Driver Download” . (2). Adding the definition of segments in "4.4.4 Reprogramming with Flash Driver Download ". (3). Adjusting the erasure, download and validity check of segment in "RS-FBL-4090 Overview of Programming Step 4 “Reprogramming with Flash Driver Download".	Release