秘密

Q/THZ

浙江合众新能源汽车有限公司企业标准

Q/THZ E8-22-2019

代替Q/THZ E8-22-2018

ECU 刷新规范

2019-11-30 发布　　2019-12-15 实施

浙江合众新能源汽车有限公司 发布

前　言

本标准依据GB/T 1.1-2009给出的规则起草。

本标准代替Q/THZ E8-22-2015《ECU刷新规范》，与Q/THZ E8-22-2015相比，除编辑性修改外，主要技术变化如下：

1）增加了擦除指令中的地址、长度信息；

2）增加了解释性说明：控制器无法响应或无法进入编程模式时，提前结束刷新。对应流程中在进入编程模式时，同步增加了相关备注。

3）流程图：取消 RID 0203 中判断条件循环处理的逻辑

4）修改了刷新流程图，增加多 sector 判断，增加芯片刷新兼容性；

5）对 8 位和 16 位校验算法进行补充说明，消除歧义。

本标准的附录 A 和附录 B 均为规范性附录。

本标准由浙江合众新能源汽车有限公司汽车工程研究院提出。

本标准由浙江合众新能源汽车有限公司产品运营中心项目管理部归口。

本标准起草单位：浙江合众新能源汽车有限公司汽车工程研究院电子电器部负责起草。

本标准主要起草人：于波、李涛。

本标准首次发布日期为 2015 年 9 月 10 日。

本标准所代替标准的历次版本发布情况为：

——Q/THZ E8-22-2015,

ECU 刷新规范

1 范围

本标准规定了UDS CAN诊断ECU刷新的基本用法及要求。

本标准适用于合众公司的乘用车。

本文档不作为UDS诊断开发规范，关于UDS诊断开发要求请参见《车辆UDS诊断CAN规范》。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

Q/THZ E8-20-2015 车辆UDS诊断CAN规范

ISO 14229:2013 Road Vehicles - Diagnostic Systems Diagnostic Services Specification

ISO 15765-2:2004 Road Vehicles-Diagnostics on Controller Area Networks (CAN) – Part2: Networking Layer Services

ISO 15765-3:2004 Road Vehicles-Diagnostics on Controller Area Networks (CAN) – Part3: Implementation of unified diagnostic services (UDS on CAN)

3 总体要求

该部分规定了合众汽车关于UDS CAN诊断ECU刷新的总体要求。

3.1 诊断服务

刷新用到的所有服务具体使用方法和要求参照合众汽车企标Q/THZ E8-20-2015：车辆UDS诊断CAN规范。

3.2 刷新文件格式

刷新文件格式只支持 S19 和 HEX 文件，不支持其他格式的文件。

3.3 刷新安全

刷新完成后，ECU 需要确保内部校验与刷新工具的校验结果一致后才能由刷新模式进入应用程序，否则仍要运行在刷新模式中，等待下一个刷新请求。

无论任何原因的刷新失败，甚至是断电， ECU 都要能再次刷新；成功刷新后的 ECU 要能再次被刷新。

针对刷新工具发出的请求（本规范定义的各条服务），若 ECU 给予不成功的响应（无响应，否定响应等），若无特别定义，刷新工具最多尝试三次请求，三次之后若 ECU 仍给予不成功的响应，则刷新设备断开通信。

4 服务与参数

本章节描述了在合众汽车定义的 ECU 刷新过程中所涉及的相关诊断服务和具体参数，其中定义了在 ECU 的应用程序软件和 Bootloader 软件中需支持的诊断服务，各诊断服务的格式满足《车辆 UDS 诊断 CAN 规范》的要求。

4.1 网络层时间参数

表1　网络层时间参数

参数	超时数值	性能要求
N_As	25ms	-
N_Ar	25ms	
N_Bs	25ms	-
N_Br	-	N_Ar + N_Br < 0.9* N_Bs
N_Cs	-	N_As + N_Cs < 0.9* N_Cr
N_Cr	150ms	-

表2　多帧传输时间参数

参数	描述	值	单位 (ms)
Bs	Block Size	0	--
STmin	Minimum Separation Time	1	ms

4.2　会话层时间参数

表3　会话层时间参数

参数	描述	类型	推荐超时值	超时值
$S3_{Client}$	诊断设备为了使多个服务器保持非默认会话模式激活状态，采用功能地址发出诊断设备在线（3E）请求的时间间隔或者为了使单个服务器保持非默认会话模式状态，而采用物理地址发出请求消息的最大间隔时间。	计时器重新载入	2000ms	5000ms
$S3_{Server}$	ECU没有收到任何诊断请求信息而保持非默认会话激活状态的时间。	计时器重新载入	N/A	5000ms

4.3　诊断服务列表

4.3.1　Bootloader 软件支持诊断服务列表

表4　Bootloader 软件支持诊断服务

诊断服务清单	会话模式	服务标识符请求信息中的值	支持禁止肯定响应	寻址		需要安全访问验证的服务
诊断服务名	刷新会话(\$02)			功能寻址	物理寻址	
诊断会话控制	√	10	√	√	√	√0
电控单元复位	√	11	-	-	√	√3
			-	√	-	√0
安全访问	√	27	-	-	√	√0
通过标识符写数据	√	2E	-	-	√	√3
例行程序控制	√	31	-	-	√	√3
请求下载	√	34	-	-	√	√3
数据传输	√	36	-	-	√	√3
请求退出传输	√	37	-	-	√	√3
待机握手	√	3E	√	√	√	√0

注：√表示该环境下支持相应诊断服务；

注：-表示该环境下不需支持相应诊断服务；

注：访问权限：√0 不需要进入安全访问权限，√1 需要扩展安全级权限，√3 需要刷新安全级权限。

4.3.2 应用程序软件支持诊断服务列表

表5 应用程序软件支持诊断服务

诊断服务清单	会话模式		服务标识符请求信息中的值	支持禁止肯定响应	寻址		需要安全访问验证的服务
诊断服务名	默认会话($01)	扩展会话($03)			功能寻址	物理寻址	
诊断会话控制	√	√	10	√	√	√	√0
电控单元复位	-	√	11	-	-	√	√1
				-	√	-	√0
清除诊断信息	√	√	14	-	√	√	√0
通过标识符读数据	√	√	22	-	-	√	√0
通讯控制	-	√	28	-	√	√	√0
例行程序控制	-	√	31	-	-	√	√0
待机握手	√	√	3E	√	√	√	√0
诊断故障码设置控制	-	√	85	-	√	√	√0
注：√表示该环境下支持相应诊断服务； 注：-表示该环境下不需支持相应诊断服务； 注：访问权限：√0 表示不需要进入安全访问权限，√1 表示需要扩展安全级权限，√3 表示需要刷新安全级权限。							

4.4 诊断会话控制$10

4.4.1 诊断会话控制请求信息

表6 诊断会话控制请求信息

数据字节	参数名称	约定	十六进制值
#1	诊断会话控制请求服务标识符	M	10
#2	诊断会话类型=[默认/刷新/扩展]	M	01/02/03

4.4.2 诊断会话控制肯定响应信息

表7 诊断会话控制肯定响应信息

数据字节	参数名称	约定	十六进制值
#1	诊断会话控制肯定响应服务标识符	M	50
#2	诊断会话类型	M	01/02/03
#3 #4	会话参数记录= P2CAN_Server_max（高字节） P2CAN_Server_max（低字节）	C	00-FF 00-FF
#5 #6	会话参数记录= P2*CAN_Server_max（高字节） P2*CAN_Server_max（低字节）	C	00-FF 00- FF

注： 字节#3-字节#6 用于诊断会话类型为刷新模式（$02）中.

4.4.3 诊断会话时间参数

刷新会话下时间参数要求如下表：

表8　诊断会话时间参数

参数	值			描述
	最小	分辨率	最大	
P2CAN_Server	0ms	1ms	25ms	刷新设备发出请求信息和收到第一帧（单帧或者多帧）之间的时间。ECU 正确接收到诊断仪发送的请求后应该在 P2CAN 时间内予以响应
P2*CAN_Server	0ms	10ms	5000ms	ECU 发出否定码为$78h 的否定应答消息与下一个响应消息（肯定或者否定响应）之间的时间
注：P2 定时器默认时间是20ms；P2*定时器默认时间是4500ms；P2*定时器精度10ms;P2定时器精度1ms。				

4.5　电控单元复位$11

4.5.1　电控单元复位请求信息

表9　电控单元复位请求信息

数据字节	参数名称	约定	十六进制值
#1	电控单元复位请求服务标识符	M	11
#2	复位类型=[硬件复位]	M	01

4.5.2　电控单元复位肯定响应信息

表10　电控单元复位肯定响应信息

数据字节	参数名称	约定	十六进制值
#1	电控单元复位肯定响应服务标识符	M	51
#2	复位类型=[硬件复位]	M	01

4.6　清除故障码 $14

4.6.1　清除故障码请求信息

表11　清除故障码请求信息

数据字节	参数名称	约定	十六进制值
#1	清除故障码请求服务标识符	M	14
	故障码类型=[		
#2	故障码类型高字节	M	FF
#3	故障码类型中间字节	M	FF
#4	故障码类型低字节]	M	FF

4.6.2　清除故障码肯定响应信息

表12　清除故障码肯定响应信息

数据字节	参数名	约定	数值
#1	清除故障码肯定响应服务标识符	M	54

4.7　通过标识符读数据 $22

4.7.1　通过标识符读数据请求信息

表13　通过标识符读数据请求信息

数据字节	参数名称	约定	十六进制值
#1	通过标识符读数据请求服务标识符	M	22
	数据标识符=[		
#2	字节#1	M	00-FF
#3	字节#2]	M	00-FF

4.7.2　通过标识符读数据肯定响应信息

表14　通过标识符读数据肯定响应信息

数据字节	参数名称	约定	十六进制值
#1	通过标识符读数据肯定响应服务标识符	M	62
	数据标识符=[		
#2	字节#1	M	00-FF
#3	字节#2]	M	00-FF
#4	数据记录=[		
:	数据#1	M	00-FF
#((k-1)+4)	:	:	:
	数据#k]	U	00-FF

4.7.3　支持的标识符

表15　支持的标识符

DID	信号/变量名称	访问权限		数据长度/字节	数据格式	数据内容
		22	2e			
F188	ECU 软件版本号（内控版本）	√0	-	8	ASCII	编码规则示例： 十六进制：30 31 2E 30 32 2E 30 33 -> ACSII：01.02.03
F199	编程日期	√0	√3	4	BCD	编码规则示例： BCD：20 16 02 02→ 2016/02/02 表示 2016 年 2 月 2 日进行 ECU 刷新
F198	维修店代码或诊断设备序列号	√0	√3	10	ASCII	
F180	Bootloader 软件版本号	√0	-	8	ASCII	编码规则示例： 30 31 2E 30 31 2E 30 31->"01.01.01" 表示 Boot 版本号为 01.01.01
F190	VIN	√0	√1	17	ASCII	编码规则示例： 十六进制：4C 56 56 44 43 31 31 42 36 41 44 35 32 34 32 38 36→ ACSII：LVVDC11B6AD324286

DID	信号/变量名称	访问权限		数据长度/字节	数据格式	数据内容
		22	2e			
F191	ECU 硬件版本号	√0	-	5	ASCII	编码规则示例： 十六进制：48 31 2E 30 31-> ACSII：H1.01 表示硬件版本号为 H1.01，
注：访问权限 √0 表示不需要安全级权限， √1 表示需要扩展安全级权限， √3 表示需要编程安全级权限，-表示不支持相应服务。						

4.8 安全访问$27

4.8.1 请求种子信息

表16 安全访问请求#1 信息

数据字节	参数名称	约定	十六进制值
#1	安全访问请求#1 服务标识符	M	27
#2	安全访问类型=请求种子	M	11

4.8.2 请求种子肯定响应信息

表17 安全访问肯定响应#1 信息

数据字节	参数名称	约定	十六进制值
#1	安全访问肯定响应#1 服务标识符	S	67
#2	安全访问类型	M	11
#3 : #n	种子 #1 (高字节) : 种子 #m (低字节)	C : C	00-FF : 00-FF

4.8.3 参数定义

安全访问服务中使用访问模式参数。它指示了电控单元执行本服务的过程，诊断设备需要安全等级进入和种子、密码的格式。

表 18 定义了请求种子和发送密码值。

表18 定义访问模式值

数值	描述
03	请求种子(在扩展会话下)
04	发送密码(在扩展会话下)
11	请求种子(在刷新会话下)
12	发送密码(在刷新会话下)

4.8.4 发送密钥请求信息

表19 安全访问请求#2 信息

数据字节	参数名称	约定	十六进制值
#1	安全访问请求#2 服务标识符	M	27
#2	安全访问类型=发送密钥	M	12
#3 : #n	密钥 #1 (高字节) : 密钥 #m(低字节)	C : C	00-FF : 00-FF

4.8.5 发送密钥肯定响应信息

表20　安全访问响应#2 信息

数据字节	参数名称	约定	十六进制值
#1	安全访问肯定响应#2 标识符	S	67
#2	安全访问类型	M	12

注：安全访问算法参照具体 ECU 的诊断参数规范。

4.9　通讯控制 $28

4.9.1　通讯控制请求信息

表21　通讯控制请求信息

数据字节	参数名	约定	十六进制值
#1	通讯控制请求服务标识符	M	28
#2	控制类型=[允许通讯 ； 禁止通讯]	M	00/ 03
#3	通讯类型=网络管理和正常通信信息	M	03

4.9.2　通讯控制肯定响应信息

表22　通讯控制肯定响应信息

数据字节	参数名	约定	十六进制值
#1	通讯控制肯定响应服务标识符	M	68
#2	控制类型=[允许通讯; 禁止通讯]	M	00/ 03

4.10　通过标识符写数据 $2E

4.10.1　通过标识符写数据请求信息

表23　通过标识符写数据请求信息

数据字节	参数名	约定	十六进制值
#1	通过标识符写数据请求服务标识符	M	2E
#2 #3	数据标识符=[字节#1(高字节) 字节#2(低字节)]	M M	00-FF 00-FF
#4 : #n	数据记录=[数据#1 : 数据#m	M : U	00-FF 00-FF 00-FF

4.10.2　通过标识符写数据肯定响应信息

表24　通过标识符写数据肯定响应信息

数据字节	参数名	约定	十六进制值
#1	通过标识符写数据肯定响应服务标识符	M	6E
#2 #3	数据标识符=[字节#1(高字节) 字节#2(低字节)]	M M	00-FF 00-FF

注：写入的数据的具体见表 15。

4.11 例行程序控制 $31

4.11.1 例行程序标识符

表25 支持的例行程序标识符

十六进制值	描述
0203	检验刷新安全条件
FF00	擦除内存
FF01	检验刷新可靠性

4.11.2 支持的例行程序状态

表26 支持的例行程序状态

数值	描述
01	诊断设备请求开始执行例行程序成功
02	例行程序完成
03	例行程序正在执行
04	停止例行程序
05	例行程序失败或者没有运行

4.11.3 检验刷新安全条件请求信息

表27 检验刷新安全条件请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型= [开始例行程序控制]	M	01
	例行程序标识符[ ] = [		
#3	例行程序状态#1 （高字节）	M	02
#4	例行程序状态#2 （低字节）]	M	03

4.11.4 检验刷新安全条件肯定响应信息

表28 检验刷新安全条件肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型= [开始例行程序控制]	M	01
	例行程序标识符[ ] = [		
#3	例行程序状态#1 （高字节）	M	02
#4	例行程序状态#2 （低字节）]	M	03
#5	例行程序控制状态= [例行程序成功 例行程序失败或者没有运行]	M	02 05

注：1.数据字节#5 中“例行程序成功”表示刷新安全条件符合；

2. 数据字节#5 中“例行程序失败或者没有运行”表示刷新安全条件不符合。

4.11.5 检验刷新安全条件状态查询请求信息

表29　　检验刷新安全条件状态查询请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型=［ 请求例行程序控制结果］	M	03
#3 #4	例行程序标识符[ ] = [例行程序状态#1 （高字节） 例行程序状态#2 （低字节）]	M M	02 03

4.11.6　检验刷新安全条件状态查询肯定响应信息

表30　检验刷新安全条件状态查询肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型=［ 请求例行程序控制结果］	M	03
#3 #4	例行程序状态记录[ ] = [例行程序状态#1 （高字节） 例行程序状态#2 （低字节）]	M M	02 03
#5	例行程序控制状态=［ 例行程序成功 例行程序正在执行 例行程序失败或者没有运行］	M	02 03 05

注：1.数据字节#5 中“例行程序成功”表示刷新安全条件符合；

2.数据字节#5 中“例行程序失败或者没有运行”表示刷新安全条件不符合。

4.11.7　擦除内存开始请求信息

表31　擦除内存开始请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型=［ 开始例行程序控制］	M	01
#3 #4	例行程序标识符[ ] = [例行程序状态#1 （高字节） 例行程序状态#2 （低字节）]	M M	FF 00
#5	Memory 地址(擦除开始的地址,byte1)(MSB)	M	00-FF
#6	Memory 地址(擦除开始的地址,byte2)	M	00-FF
#7	Memory 地址(擦除开始的地址,byte3)	M	00-FF
#8	Memory 地址(擦除开始的地址,byte4)（LSB）	M	00-FF
#9	Memory 长度(擦除的长度信息,byte1) (MSB)	M	00-FF
#10	Memory 长度(擦除的长度信息,byte2)	M	00-FF
#11	Memory 长度(擦除的长度信息,byte3)	M	00-FF
#12	Memory 长度(擦除的长度信息,byte4) （LSB）	M	00-FF

4.11.8 擦除内存开始肯定响应信息

表32 擦除内存开始肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型= [开始例行程序控制]	M	01
	例行程序状态记录[ ] = [		
#3	例行程序状态#1（高字节）	M	FF
#4	例行程序状态#2（低字节）]	M	00
#5	例行程序控制状态= [例行程序成功 例行程序正在执行 例行程序失败或者没有运行]	M	 02 03 05

注:1.由于擦除内存需要一定时间，故允许 ECU 在回复肯定响应之前，连续反馈若干个 NRC78 否定响应码。

2.数据字节#5 中“例行程序成功”表示擦除内存成功；

3.数据字节#5中“例行程序失败或者没有运行”表示擦除内存不成功。

4.11.9 擦除内存状态查询请求信息

表33 擦除内存状态查询请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型= [请求例行程序控制结果]	M	03
	例行程序标识符[ ] = [		
#3	例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	00

4.11.10 擦除内存状态查询肯定响应信息

表34 擦除内存状态查询肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型= [请求例行程序控制结果]	M	03
	例行程序状态记录[ ] = [		
#3	例行程序状态#1（高字节）	M	FF
#4	例行程序状态#2（低字节）]	M	00
#5	例行程序控制状态= [例行程序成功 例行程序正在执行 例行程序失败或者没有运行]	M	 02 03 05

注：1.数据字节#5 中“例行程序成功”表示擦除内存成功；

2.数据字节#5 中“例行程序失败或者没有运行”表示擦除内存不成功

4.11.11 擦除内存停止请求信息

表35 擦除内存停止请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型= [请求例行程序控制结果]	M	02
	例行程序标识符[ ] = [		
#3	例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	00

4.11.12 擦除内存停止肯定响应信息

表36 擦除内存停止肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型= [请求例行程序控制结果]	M	02
	例行程序状态记录[ ] = [		
#3	例行程序状态#1（高字节）	M	FF
#4	例行程序状态#2（低字节）]	M	00
	例行程序控制状态= [		
#5	停止例行程序	M	04
	例行程序失败或者没有运行]		05

4.11.13 检验刷新可靠性开始请求信息

表37 检验刷新可靠性开始请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型= [开始例行程序控制]	M	01
	例行程序标识符[ ] = [		
#3	例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	01
#5	Memory 地址(擦除开始的地址,byte1)(MSB)	M	00-FF
#6	Memory 地址(擦除开始的地址,byte2)	M	00-FF
#7	Memory 地址(擦除开始的地址,byte3)	M	00-FF
#8	Memory 地址(擦除开始的地址,byte4)（LSB）	M	00-FF
#9	Memory 长度(擦除的长度信息,byte1) (MSB)	M	00-FF
#10	Memory 长度(擦除的长度信息,byte2)	M	00-FF
#11	Memory 长度(擦除的长度信息,byte3)	M	00-FF
#12	Memory 长度(擦除的长度信息,byte4) （LSB）	M	00-FF
#13	校验和字节 #1(高字节)	M	00-FF
#14	校验和字节 #2 (低字节)]		00-FF

4.11.14 检验刷新可靠性开始肯定响应信息

表38 检验刷新可靠性开始肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型= [开始例行程序控制]	M	01
	例行程序状态记录[ ] = [		
#3	例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	01
#5	例行程序控制状态= [例行程序成功 例行程序失败或者没有运行]	M	02 05
#6	校验和字节 #1(高字节)	M	00-FF
#7	校验和字节 #2 (低字节)]		00-FF

注:1.由于计算校验和可能需要一定时间，故允许 ECU 在回复肯定响应之前，连续反馈若干个 NRC78 否定响应码

2.数据字节#5 中“例行程序成功”表示 ECU 校验的结果和诊断仪校验的结果一致；

3.数据字节#5 中“例行程序失败或者没有运行”表示 ECU 校验的结果和诊断仪校验的结果不一致

4.11.15 检验刷新可靠性状态查询请求信息

表39 检验刷新可靠性状态查询请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型= [请求例行程序控制结果]	M	03
	例行程序标识符[ ] = [		
#3	例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	01

4.11.16 检验刷新可靠性状态查询肯定响应信息

表40 检验刷新可靠性状态查询肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型= [请求例行程序控制结果]	M	03
	例行程序状态记录[ ] = [		
#3	例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	01
#5	例行程序控制状态= [例行程序成功 例行程序正在执行 停止例行程序 例行程序失败或者没有运行]	M	02 03 04 05

4.11.17 停止检验刷新可靠性请求信息

表41　停止检验刷新可靠性请求信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务请求标识符	M	31
#2	例行程序控制类型= [请求例行程序控制结果]	M	02
#3	例行程序标识符[ ] = [例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	01

4.11.18　停止检验刷新可靠性肯定响应信息

表42　停止检验刷新可靠性肯定响应信息

数据字节	参数名	约定	十六进制值
#1	例行程序服务肯定响应标识符	M	71
#2	例行程序控制类型= [请求例行程序控制结果]	M	02
#3	例行程序状态记录[ ] = [例行程序状态#1 （高字节）	M	FF
#4	例行程序状态#2 （低字节）]	M	01
#5	例行程序控制状态= [停止例行程序 例行程序失败或者没有运行]	M	04 05
#6	例行程序状态记录 [] = [校验和字节 #1(高字节)	M	00-FF
#7	校验和字节 #2 (低字节)]		00-FF

注：请参考附录 A:CRC16 校验和算法

4.12　请求下载 $34

4.12.1　请求下载请求信息

表43　请求下载服务请求信息

数据字节	参数名	约定	十六进制值
#1	请求下载服务标识符	M	34
#2	数据格式标识符[] = [不压缩；不加密]	M	00
#3	地址长度格式标识符[] = [内存 4GB；内存地址 4GB]	M	44
#4	内存地址　[字节#1] (高字节)	M	00-FF
#5	[字节#2]	M	00-FF
#6	[字节#3]	M	00-FF
#7	[字节#4] (低字节)	M	00-FF
#8	内存大小　[字节#1] (高字节)	M	00-FF
#9	[字节#2]	M	00-FF
#10	[字节#3]	M	00-FF
#11	[字节#4] (低字节)	M	00-FF

4.12.2 请求下载肯定响应信息

表44 请求下载服务肯定响应信息

数据字节	参数名	约定	十六进制值
#1	请求下载肯定响应服务标识符	M	74
#2	长度格式标识符	M	20
#3	最大块长度数量[字节#1]	M	00-FF
#4	最大块长度数量[字节#2]	M	00-FF

4.13 传输数据$36

4.13.1 传输数据请求信息

表45 数据传输请求信息

数据字节	参数名	约定	十六进制值
#1	传输数据请求服务标识符	M	36
#2	块顺序计数器	M	00-FF
#3 : #n	传输请求参数记录[] = [传送的数据#1 : 传送的数据#m	C* : U	00-FF : 00-FF

表46 数据传输肯定响应信息

数据字节	参数名	约定	十六进制值
#1	传输数据请求肯定响应服务标识符	M	76
#2	块顺序计数器	M	00-FF

4.14 传输数据请求传输退出$37

4.14.1 传输数据请求信息

表47 请求传输退出请求信息

数据字节	参数名	约定	十六进制值
#1	请求传输退出请求服务标识符	M	37

4.14.2 传输数据肯定响应信息

表48 请求传输退出肯定响应信息

数据字节	参数名	约定	十六进制值
#1	请求数据传输退出肯定响应服务标识符	M	77
#2	请求退出响应参数记录[] = [校验和字节]	M	00-FF

注：请参考附录B:CRC8校验和算法。

4.15 链路保持 $3E

4.15.1 链路保持请求信息

表49 链路保持请求信息

数据字节	参数名称	约定	十六进制值
#1	链路保持请求服务标识符	M	3E
#2	零子功能	M	80

注:整个刷新过程中，刷新工具要周期性的发送链路保持请求，ECU 不需要响应请求信息。

4.16 诊断故障代码设置控制 $85

4.16.1 诊断故障代码设置控制请求信息

表50 诊断故障码设置控制请求信息

数据字节	参数名称	约定	十六进制值
#1	诊断故障码设置控制请求服务标识符	M	85
#2	故障码设置类型[] = [On/Off]	M	01/02

4.16.2 诊断故障代码设置控制肯定响应信息

表51 诊断故障码设置控制肯定响应信息

数据字节	参数名称	约定	十六进制值
#1	诊断故障码设置控制肯定响应服务标识符	M	C5
#2	故障码设置类型[] = [On/Off]	M	01/02

4.17 链路控制 $87

4.17.1 确认波特率请求信息

表52 确认波特率请求信息

数据字节	参数名称	约定	十六进制值
#1	链路控制请求服务标识符	M	87
#2	链路控制类型= [确认转换到指定波特率]	M	02
#3	链路波特率记录 [波特率高字节]	M	00-FF
#4	链路波特率记录 [波特率中间字节]	M	00-FF
#5	链路波特率记录 [波特率低字节]	M	00-FF

4.17.2 确认波特率肯定响应信息

表53 确认波特率肯定响应信息

数据字节	参数名称	约定	十六进制值
#1	链路控制肯定响应服务标识符	M	C7
#2	链路控制类型	M	02

4.17.3 转换波特率请求信息

表54 转换波特率请求信息

数据字节	参数名称	约定	十六进制值
#1	链路控制请求服务标识符	M	87
#2	链路控制类型= [转换波特率]	M	83
#3	链路波特率记录 [波特率高字节]	M	00-FF
#4	链路波特率记录 [波特率中间字节]	M	00-FF
#5	链路波特率记录 [波特率低字节]	M	00-FF

5 ECU 刷新流程

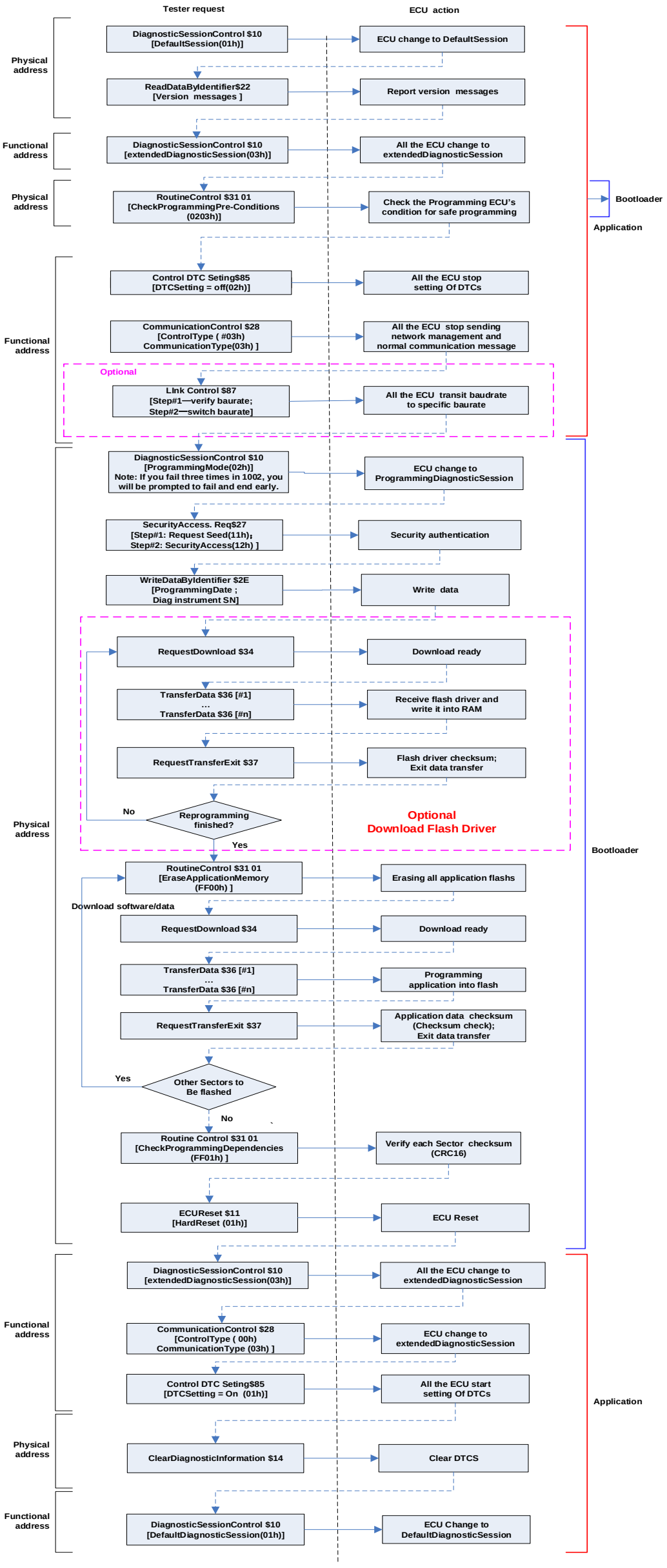
Tester request
ECU action
Physical address
DiagnosticSessionControl $10 [DefaultSession(01h)]
ECU change to DefaultSession
ReadDataByIdentifier$22 [Version messages]
Report version messages
Functional address
DiagnosticSessionControl $10 [extendedDiagnosticSession(03h)]
All the ECU change to extendedDiagnosticSession
Physical address
RoutineControl $31 01 [CheckProgrammingPre-Conditions (0203h)]
Check the Programming ECU's condition for safe programming
Bootloader
Application
Functional address
Control DTC Seting$85 [DTCSetting = off(02h)]
All the ECU stop setting Of DTCs
CommunicationControl $28 [ControlType (#03h) CommunicationType(03h)]
All the ECU stop sending network management and normal communication message
Optional
Link Control $87 [Step#1—verify baurate; Step#2—switch baurate]
All the ECU transit baudrate to specific baurate
DiagnosticSessionControl $10 [ProgrammingMode(02h)] Note: If you fail three times in 1002, you will be prompted to fail and end early.
ECU change to ProgrammingDiagnosticSession
SecurityAccess. Req$27 [Step#1: Request Seed(11h)， Step#2: SecurityAccess(12h)]
Security authentication
WriteDataByIdentifier $2E [ProgrammingDate ; Diag instrument SN]
Write data
RequestDownload $34
Download ready
TransferData $36 [#1] ... TransferData $36 [#n]
Receive flash driver and write it into RAM
RequestTransferExit $37
Flash driver checksum; Exit data transfer
No
Reprogramming finished?
Yes
Optional
Download Flash Driver
Physical address
Bootloader
RoutineControl $31 01 [EraseApplicationMemory (FF00h)]
Erasing all application flashs
Download software/data
RequestDownload $34
Download ready
TransferData $36 [#1] ... TransferData $36 [#n]
Programming application into flash
RequestTransferExit $37
Application data checksum (Checksum check); Exit data transfer
Yes
Other Sectors to Be flashed
No
Routine Control $31 01 [CheckProgrammingDependencies (FF01h)]
Verify each Sector checksum (CRC16)
ECUReset $11 [HardReset (01h)]
ECU Reset
Functional address
DiagnosticSessionControl $10 [extendedDiagnosticSession(03h)]
All the ECU change to extendedDiagnosticSession
CommunicationControl $28 [ControlType (00h) CommunicationType (03h)]
ECU change to extendedDiagnosticSession
Control DTC Seting$85 [DTCSetting = On (01h)]
All the ECU start setting Of DTCs
Application
Physical address
ClearDiagnosticInformation $14
Clear DTCS
Functional address
DiagnosticSessionControl $10 [DefaultDiagnosticSession(01h)]
ECU Change to DefaultDiagnosticSession

图1

6 ECU 刷新过程

6.1 概述

本章描述了刷新应用软件/数据到 ECU 内存区域流程的框架结构。刷新过程被分成三个阶段：刷新准备阶段（设置刷新网络），刷新阶段（下载软件/数据）和刷新后的处理阶段。

根据刷新过程的具体需求，分别定义了以上三个阶段使用的寻址方式（物理地址或者功能地址），

刷新设备遵循以下刷新流程，负责将正确的刷新文件（S19 或者 HEX）下载到 ECU 中。

对于要求实现 OTA 的 ECU，在刷新失败后，应能实现应用程序回滚的功能。

刷新过程概述如下图：

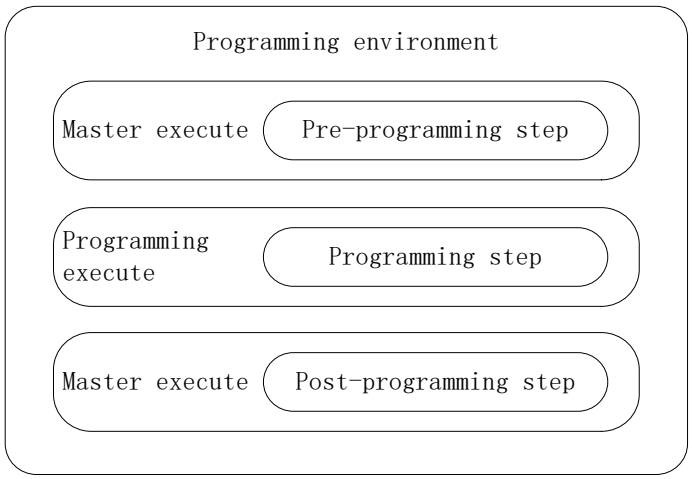

图2

6.2 刷新准备阶段

刷新准备阶段需要确认待刷新控制器的相关版本信息，设置刷新网络等。这个阶段在整车各个控制器的应用程序中执行，此阶段，使用功能地址向网络上的各控制器发出诊断请求进行网络设置（步骤 a – 步骤 g）。具体流程如下图所示：

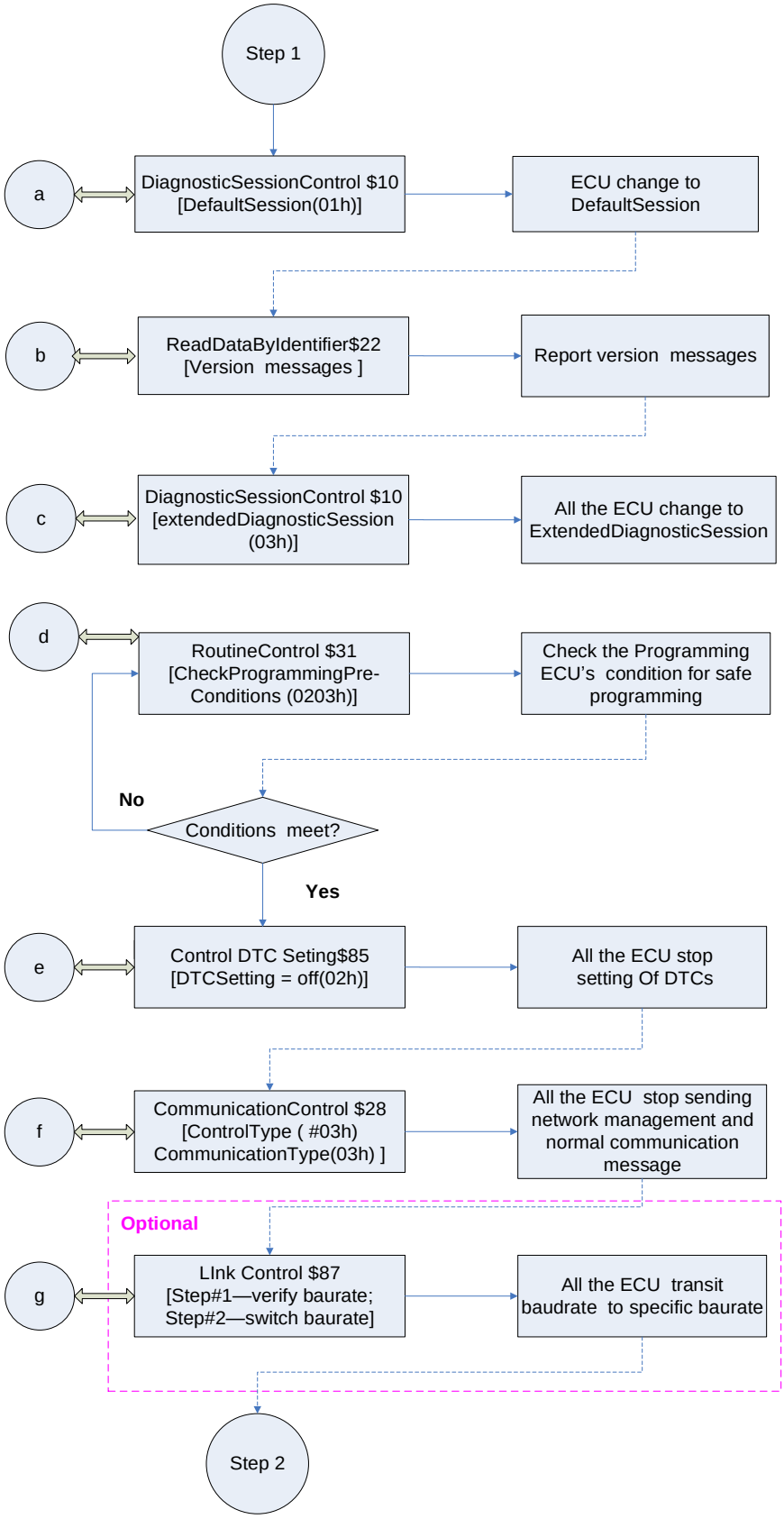

图3

a）进入默认会话模式；

b) ECU 相关信息确认。

刷新前，刷新工具读取 ECU 的 Boot 软件版本号（F180）、软件版本（F188）、VIN(F190)、硬件版本（F191），根据从 ECU 获取到的相关信息到刷新数据库中查找对应的升级文件。维修店代码或诊断设备序列号（F198）、刷新日期（F199）在刷新启动时写入，用于追溯之前的刷新操作。

c) 在设置刷新网络前，ECU 需要进入扩展模式。在扩展模式下，DTC 检测和非诊断消息发送被禁止。具体消息格式见 4.4.1 节。

为了保持网络上所有控制器处于扩展模式,刷新设备需要周期性以功能地址方式发送链路保持命令（TesterPresent），具体格式见 4.15 节。

d) 该例程标识符允许客户端检测转换到刷新会话的所有刷新安全条件是否满足。此步骤不需要经过安全访问算法保护。且要求 ECU 的 Application 和 Bootloader 中均可对此请求信息给予肯定响应。

具体消息格式见 4.11.3 节

需要检查的刷新安全条件包括如下：

- ECU 的电源电压不能太高或者太低（9V-16V）；
- 车辆处于 IGN On 状态，但不在 Ready 状态；
- 车辆处于静止状态，车速为 0km/h；

注:网络管理节点在 IGN Off 状态下也可执行刷新操作。

如果刷新安全条件通过，ECU 给予肯定响应。刷新设备将继续后面的刷新流程。如果 ECU 出厂时没有应用软件，只有 Bootloader，ECU 将不需要做任何的检查而直接肯定响应该请求信息。

如果检验刷新安全条件的例行程序执行失败，刷新流程断开，需要人工进行检查失败原因。

e) 刷新期间，要求各 ECU 停止故障码的检测。具体消息格式见 4.16 节。

说明：本步骤不作为刷新流程异常断开的判断条件。

f) 刷新期间，要求各个 ECU 停止非诊断消息的发送及接收，具体消息格式见 4.9 节。

说明：本步骤不作为刷新流程异常断开的判断条件。

g) 在生产线上，为了减少刷新执行时间，可以适当提高刷新的波特率。提高波特率根据应用需求作为可选项。具体实现步骤如下：

步骤 1: 确认 ECU 能执行波特率改变。ECU 在成功响应该请求后，波特率并未改变。具体消息格式见 4.17.1 节；

步骤 2：改变波特率。一旦 ECU 成功响应该请求后，刷新设备和 ECU 同时改变到预先设定的波特率。具体消息格式见 4.17.3 节。

在改变到预先设定的波特率前，ECU 必须设置一个特定的时间窗口，在时间窗口内刷新设备不允许发送任何请求（包括服务 TesterPresent ($3E)）。波特率一旦被成功的改变后，在整个扩展模式下均保持改变后的波特率，一旦 ECU 回到默认会话模式，波特率也将恢复到改变前的状态。

说明：本步骤不作为刷新流程异常断开的判断条件。

6.3 刷新阶段：下载数据

该阶段在 Bootloader 中执行，诊断消息寻址方式均为物理地址。刷新步骤如下：

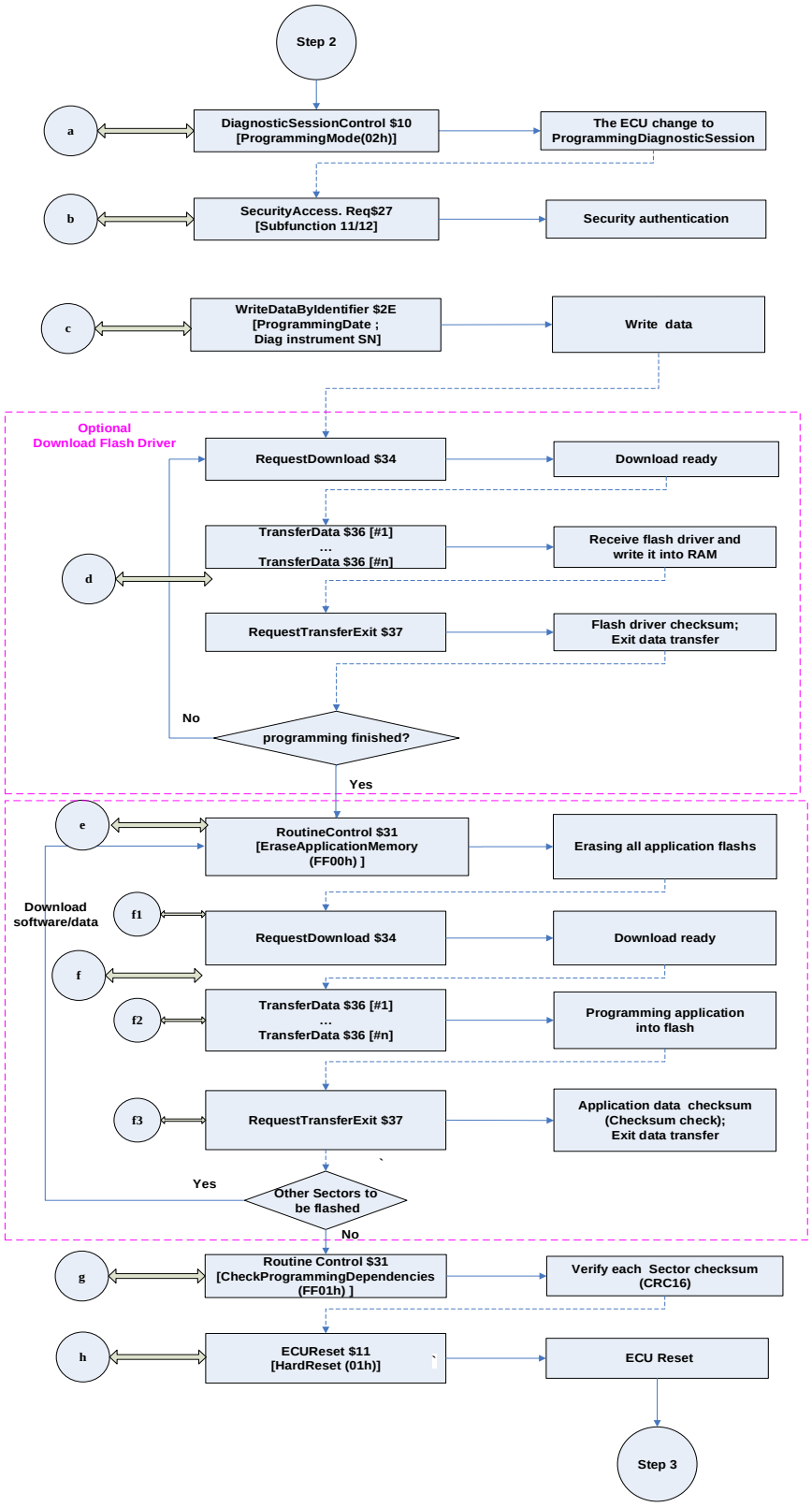

图4

a）ECU 刷新相关操作需要在刷新会话模式下进行，ECU 进入该模式后，会为即将到来的刷新准备好所有

必要的资源。如果出现下位机无法响应编程模式请求或者进入不了编程模式的（连续 3 次请求均失败），上位机停止请求并提前结束刷新流程，提示刷新失败。

b) 刷新事件应该是安全的，因此，对于刷新，安全访问服务($27) 是必须的。具体消息格式见 4.8 节。

c) 通过安全解锁后，可以写入刷新的信息，如：维修店代码或诊断设备序列号(F198)和刷新日期(F199)等。具体消息格式见 4.10 节

d) 如果 ECU 内存区域内没有刷新驱动程序（内存清除例行程序），需要下载刷新驱动程序。下载过程将按照特定的顺序执行请求下载 ($34)，传输数据 ($36)，请求传输退出($37)服务。具体消息格式见 4.12-4.14 节。

e) 在向 ECU 的内存区域下载数据之前，需要先擦除内存区域已有数据。本步骤将根据控制器地址空间分配和芯片擦除能力，区分单次擦除所有或多次分段擦除的形式。擦除内存的具体消息格式见 4.11.7 节。

需要额外指出的是：

- 多次擦除需要在每一次写入并校验成功的情况下进行；
- 8 位校验和算法仅对单次有效，不进行累加；
- 16 位校验和算法为整个 S19 的校验，多段时为累加运算。

例行程序的结果需要在例行程序状态字节中反馈回来，用于刷新设备判断是否执行成功以及失败原因。

f) 每次下载一个应用软件/数据的数据块到 ECU 的内存中都需要遵循以下服务顺序：

- 请求下载 ($34)；
- 数据出传输 ($36)；
- 请求传输退出 ($37)；

1) 刷新设备使用服务请求下载 ($34)向 ECU 指定刷新起始地址和刷新数据的大小,请求下载 ($34) 服务指定的内存从起始到结束应该是连续的。刷新设备应该为每个要刷新的数据块发送一个单独的请求服务。

2) 刷新设备使用服务传输数据 ($36) 向 ECU 内存区域中传输要刷新的数据，参数块顺序计数器被用来作为数据块计数器去保证正确的数据块传输。一旦计数器错误，ECU 将等待新的传输。一个单独的数据块可能需要多条传输数据 ($36) 服务请求信息去传输（数据块的长度超过了网络层允许的最大缓冲区域）。

3) 刷新设备使用服务请求传输退出 ($37) 退出当前的连续内存区域的刷新。Bootloader 将在请求传输退出 ($37) 服务肯定响应中返回 1 字节的校验和校验，校验的范围是最近的一条请求下载 ($34) 服务指定的内存区域。该校验和字节被刷新设备用来验证数据传输的准确性。具体消息格式见 4.14 节

一旦校验和校验错误，刷新设备将重复相同的数据传输($36)请求，包括相同的块顺序计数器，多次校验仍不能通过，刷新将终止。

关于校验和校验的算法参考附录 B。

如果有几个内存区域要刷新，刷新设备将发送多个请求下载($34)；传输数据($36)及请求传输退出($37) 循环顺序。

g) 采用单次擦除方式的，在软件/数据刷新完毕时，刷新设备通过例行程序服务来验证刷新到内存区域的每块数据是否成功。具体消息格式见 4.11.13 节；采用多次分段擦写方式时，在通过 37 正响应后，应判断 sector 是否刷新完毕。当判断结果为否时，需请求擦除下一 secotr。

该例行程序触发 ECU 执行 CRC16 校验，ECU 通过肯定响应中的 2 字节的 CRC 校验参数将校验结果反馈刷新设备。如果 CRC 校验失败，刷新设备将终止刷新流程。CRC 校验方法参看附录 A。

刷新设备在请求服务信息(例行程序控制可选记录) 中将校验和下发给 ECU,ECU 也将该校验和与刷新设备计算的校验和相比较用于判断该块刷新是否成功。

ECU 将例行程序的结果体现到例行程序状态字节中并反馈给刷新设备，刷新设备根据反馈信息判断是否执行成功以及失败原因。

h）整个刷新完成后，刷新设备要求 ECU 硬件复位，ECU 进入应用程序。此时网络恢复到正常的模式，ECU 以默认的波特率进行正常的通信，并能进行故障码的检测和存储。

刷新后处理

该阶段是复位后在应用程序中执行一些恢复整车网络通信的操作。

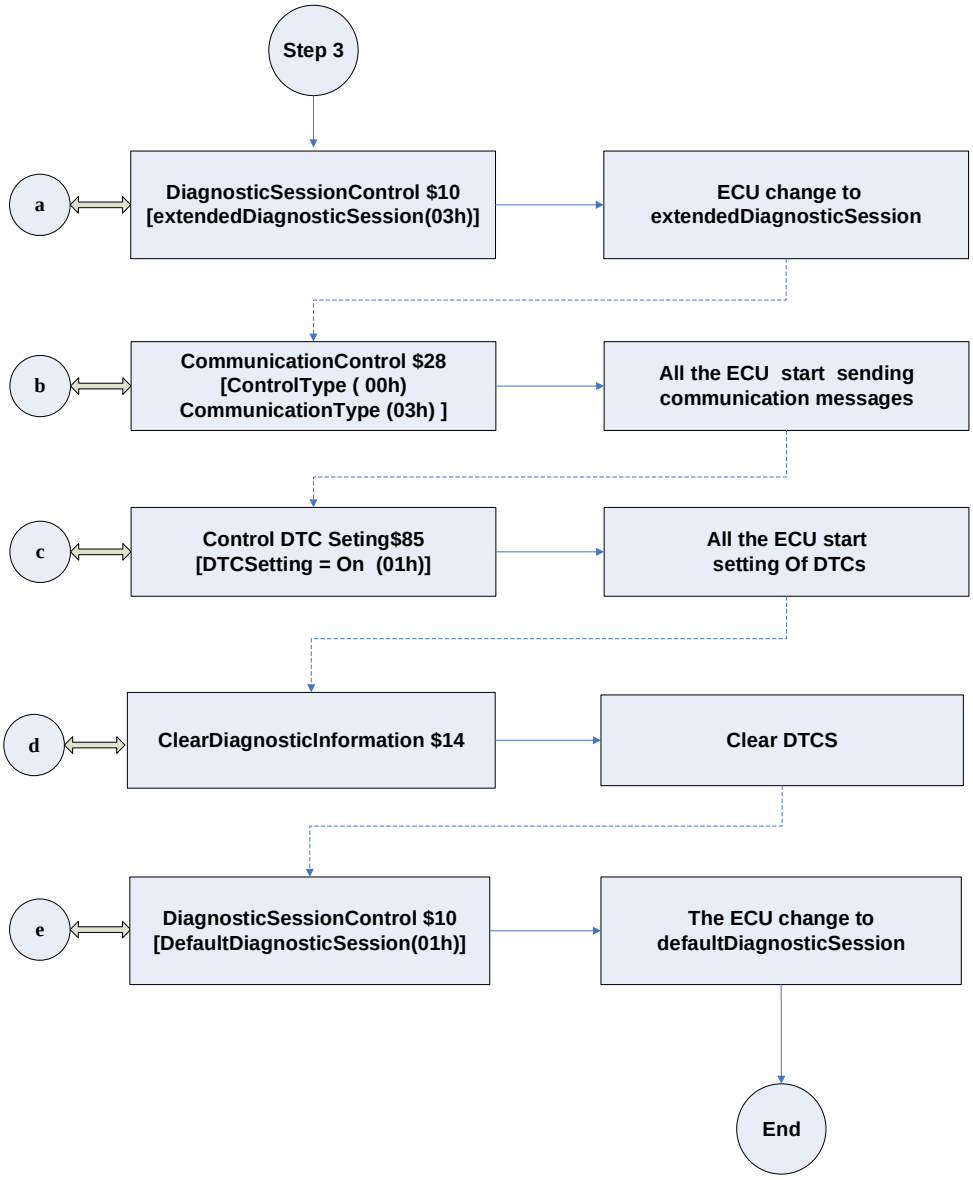

图5

a） 进入扩展模式。

b）刷新结束后要求各 ECU 恢复非诊断消息的发送及接收。

说明：本步骤不作为刷新流程异常断开的判断条件。

c）要求各 ECU 恢复故障码的检测。

说明：本步骤不作为刷新流程异常断开的判断条件。

d）在刷新过程中，ECU 可能潜在的产生一些故障码，清除故障码服务($14 h)被用来清除故障码。

e）ECU 回到默认模式。

7 服务器刷新要求

7.1 Boot 软件通用要求

所有支持应用程序刷新的可编程服务器都应在 Boot 内存分区中包含 Boot 软件。上电时，服务器首先运行 Boot 软件。服务器收到刷新请求时也会进入 Boot 软件。下图描述了 Boot 软件和应用软件之间的交互和转换关系。

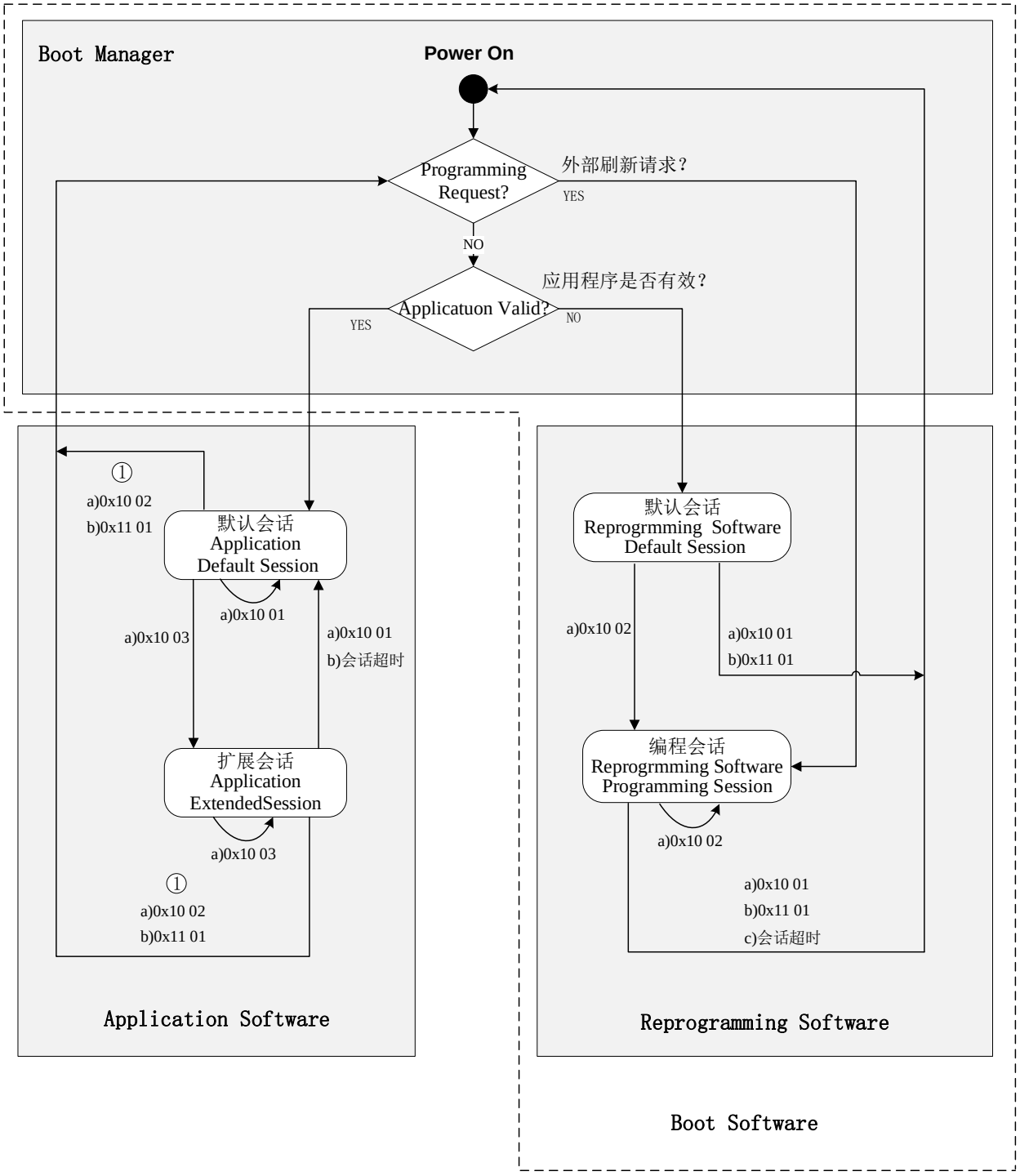

图6

附录 A

（规范性附录）

CRC16-CITT 校验和算法

CRC16-CITT 校验和算法（以 16 位单片机为例）

The checksum algorithm to be used shall be the CRC16-CITT:

- Polynomial: x^16+x^12+x^5+1 (1021 hex)
- Initial value: FFFF (hex)

For a fast CRC16-CITT calculation a look-up table implementation is the preferred solution. For ECUs with a limited amount of flash memory (or RAM), other implementations may be necessary.

Example 1: crc16-citt c-code (fast)

This example uses a look-up table with pre-calculated CRCs for fast calculation.

```
/* function declarations */
unsigned short int CalcCRC(unsigned short int size, unsigned char *data);
/*Here is the test data array ,the length of the data array depends on the actual data array */
unsigned char data[256] =
{
    0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,
    0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,
    0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,
    0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x3c,0x3d,0x3e,0x3f,
    0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,0x4c,0x4d,0x4e,0x4f,
    0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5a,0x5b,0x5c,0x5d,0x5e,0x5f,
    0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6a,0x6b,0x6c,0x6d,0x6e,0x6f,
    0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7a,0x7b,0x7c,0x7d,0x7e,0x7f,
    0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8a,0x8b,0x8c,0x8d,0x8e,0x8f,
    0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9a,0x9b,0x9c,0x9d,0x9e,0x9f,
    0xa0,0xa1,0xa2,0xa3,0xa4,0xa5,0xa6,0xa7,0xa8,0xa9,0xaa,0xab,0xac,0xad,0xae,0xaf,
    0xb0,0xb1,0xb2,0xb3,0xb4,0xb5,0xb6,0xb7,0xb8,0xb9,0xba,0xbb,0xbc,0xbd,0xbe,0xbf,
    0xc0,0xc1,0xc2,0xc3,0xc4,0xc5,0xc6,0xc7,0xc8,0xc9,0xca,0xcb,0xcc,0xcd,0xce,0xcf,
    0xd0,0xd1,0xd2,0xd3,0xd4,0xd5,0xd6,0xd7,0xd8,0xd9,0xda,0xdb,0xdc,0xdd,0xde,0xdf,
    0xe0,0xe1,0xe2,0xe3,0xe4,0xe5,0xe6,0xe7,0xe8,0xe9,0xea,0xeb,0xec,0xed,0xee,0xef,
    0xf0,0xf1,0xf2,0xf3,0xf4,0xf5,0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfd,0xfe,0xff
};
    /*Here is crctab[256], this array is fixed */
unsigned short int crctab[256] =
{
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
    0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52B5, 0x4294, 0x72F7, 0x62D6,
    0x9339, 0x8318, 0xB37B, 0xA35A, 0xD3BD, 0xC39C, 0xF3FF, 0xE3DE,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64E6, 0x74C7, 0x44A4, 0x5485,
    0xA56A, 0xB54B, 0x8528, 0x9509, 0xE5EE, 0xF5CF, 0xC5AC, 0xD58D,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76D7, 0x66F6, 0x5695, 0x46B4,
```

```
    0xB75B, 0xA77A, 0x9719, 0x8738, 0xF7DF, 0xE7FE, 0xD79D, 0xC7BC,
    0x48C4, 0x58E5, 0x6886, 0x78A7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xC9CC, 0xD9ED, 0xE98E, 0xF9AF, 0x8948, 0x9969, 0xA90A, 0xB92B,
    0x5AF5, 0x4AD4, 0x7AB7, 0x6A96, 0x1A71, 0x0A50, 0x3A33, 0x2A12,
    0xDBFD, 0xCBDC, 0xFBBF, 0xEB9E, 0x9B79, 0x8B58, 0xBB3B, 0xAB1A,
    0x6CA6, 0x7C87, 0x4CE4, 0x5CC5, 0x2C22, 0x3C03, 0x0C60, 0x1C41,
    0xEDAE, 0xFD8F, 0xCDEC, 0xDDCD, 0xAD2A, 0xBD0B, 0x8D68, 0x9D49,
    0x7E97, 0x6EB6, 0x5ED5, 0x4EF4, 0x3E13, 0x2E32, 0x1E51, 0x0E70,
    0xFF9F, 0xEFBE, 0xDFDD, 0xCFFC, 0xBF1B, 0xAF3A, 0x9F59, 0x8F78,
    0x9188, 0x81A9, 0xB1CA, 0xA1EB, 0xD10C, 0xC12D, 0xF14E, 0xE16F,
    0x1080, 0x00A1, 0x30C2, 0x20E3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83B9, 0x9398, 0xA3FB, 0xB3DA, 0xC33D, 0xD31C, 0xE37F, 0xF35E,
    0x02B1, 0x1290, 0x22F3, 0x32D2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xB5EA, 0xA5CB, 0x95A8, 0x8589, 0xF56E, 0xE54F, 0xD52C, 0xC50D,
    0x34E2, 0x24C3, 0x14A0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xA7DB, 0xB7FA, 0x8799, 0x97B8, 0xE75F, 0xF77E, 0xC71D, 0xD73C,
    0x26D3, 0x36F2, 0x0691, 0x16B0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xD94C, 0xC96D, 0xF90E, 0xE92F, 0x99C8, 0x89E9, 0xB98A, 0xA9AB,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18C0, 0x08E1, 0x3882, 0x28A3,
    0xCB7D, 0xDB5C, 0xEB3F, 0xFB1E, 0x8BF9, 0x9BD8, 0xABBB, 0xBB9A,
    0x4A75, 0x5A54, 0x6A37, 0x7A16, 0x0AF1, 0x1AD0, 0x2AB3, 0x3A92,
    0xFD2E, 0xED0F, 0xDD6C, 0xCD4D, 0xBDAA, 0xAD8B, 0x9DE8, 0x8DC9,
    0x7C26, 0x6C07, 0x5C64, 0x4C45, 0x3CA2, 0x2C83, 0x1CE0, 0x0CC1,
    0xEF1F, 0xFF3E, 0xCF5D, 0xDF7C, 0xAF9B, 0xBFBA, 0x8FD9, 0x9FF8,
    0x6E17, 0x7E36, 0x4E55, 0x5E74, 0x2E93, 0x3EB2, 0x0ED1, 0x1EF0
};
/* Example: code to calculate checksum*/

unsigned short int crc=0xffff;    /* global variable */

void main(void)        /* driver */
{
    unsigned short int result_crc;
    result_crc=CalcCRC(256,data);
    while(1); /* the result of the above calculation shall be: crc=0x3FBD */
}
/* Calculate CRC */
unsigned short int CalcCRC(unsigned short int size, unsigned char *data)
{
    unsigned short int tmp;
    unsigned short int i;
    for(i=0;i<size;i++)
   {
        tmp=(crc>>8)^data[i];
```

```
        crc=(crc<<8)^crctab[tmp];
    }
    return crc;
}
```

Note: This algorithm example can apply to calculate the CRC of the whole block by making every segment of the block take part in the arithmetic.

附录 B

（规范性附录）
8位校验和校验算法

在请求数据传输退出服务响应信息中，传输响应记录参数里面包含 1 字节的校验和校验结果。该 1 字节的校验和校验在数据刷新到 ECU 内存区域后进行校验，校验范围是最近的一条请求下载服务（$34）中指定的数据大小。

Example : 8-bit Checksum algorithm

```
/* function declarations */
unsigned short int CalcCRC(unsigned short int size, unsigned char *data);
unsigned short int tmp=0x00;     /* global variable */

/* test data array */
unsigned char data[32] =
{
    0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,
    0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f
};

/* Example :code to calculate checksum*/

void main(void) /* driver */
{
    unsigned short int calc;
    unsigned char crc;
    calc =CalcCRC(32, data);

/* Since crc is char type, the high byte of tmp is ignored and crc shall be 1‘s complement of low byte */

    crc =~ ((char) calc);
    while(1); /* the result of the above calculation shall be: crc=0x0F */
}

/* Calculate CRC */
unsigned short int CalcCRC(unsigned short int size, unsigned char *data)
{
    unsigned short int i;
    for(i=0;i<size;i++)
    {
        tmp= tmp + data[i];
    } /* result of above calculation shall be: tmp=0x1F0 */
    return tmp;
}
```

编制：

校对：

审核：

标准化：

批准：

版本：01