D3 项目 Bootloader 需求规范

目　　录

更新历史

版本	编写/修改	日期	内容描述	文档状态（Draft/Release）
V0.1	郭俊飞、娄鹏辉	2015.12.25	草版第一版	Draft

1 范围

本文件规定了知豆电动汽车有限公司 D3 车型整车 Bootloader 需求规范。

本文件作为各 ECU 技术要求的组成部分，各 ECU 供应商应严格遵守。

本文件是对 UDS 相关 ISO 标准的具体化，本文中没有提到的需求请参照相关 ISO 标准。

2 标准参考

下列文件中的条款通过本文件的引用成为本文件的条款。凡是注日期的引用文件，其随后的修改单（不包括勘误的内容）或修订版均不适用于本标准。凡是不注日期的引用文件，其最新版本适用于本文件。

[1] D3 项目诊断需求规范
[2] ISO 14229-1: 2006
[3] ISO 15765-2: 2004
[4] ISO 15765-3: 2004

3 缩略语

简写	英文	中文
ECU	Electronic Control Unit	电子控制单元

4 通用需求

4.1 概述

所有支持应用程序及应用程序数据编程的 ECU，应当包含 Bootloader 程序。在 ECU 正常运行过程中，执行的是应用程序和应用序数据。仅当应用程序或应用序数据失效，或者要求对此进行升级的时候，Bootloader 程序才被激活。

应用程序和应用程序数据可以同时编程或者相互独立编程。

4.2 硬件要求

可编程的 ECU 必须提供足够的 RAM 空间来维持下载。

4.3 软件要求

Bootloader 程序必须储存在受保护的存储器中，从而确保即使存在潜在错误，ECU 始终是可编程的。

如果没有有效的应用程序，Bootloader 程序应将 I/O 端口设置到一个安全的状态，从而避免车辆或操作人员出现危险。为了使 ECU 进入低功率状态，推荐 I/O 端口设置成最小功耗状态。

4.4 安全要求

针对如下情况，ECU 应满足一些安全要求，以确保编程进程：

◆未经授权的软件编程动作；
◆已经下载错误的软件到 ECU 内。

4.4.1 安全访问

所有可编程的 ECU 应该支持$27 安全访问服务，从而保护 ECU 免遭未经授权的软件编程操作影响。
安全访问服务子功能 07 与 08 用于编程会话模式，详见参考标准[1]。

4.4.2 预编程条件

ECU 应该确保编程的执行处于安全状态。如果预编程编程条件不满足（如车辆正在行驶等），编程请求应被拒绝。
为满足此需求，定义了一个特定的例程控制，详细信息参考第 5 章。

4.4.3 完整性验证

ECU 应该检查已下载到存储器中数据的完整性。当一个逻辑块下载后，将使用 CRC-32 算法验证当前块的所有数据字节是否被正确传输和下载。该验证通过检查编程完整性例程控制服务来激活，当接收到此例程控制服务请求时，Bootloader 将计算下载数据字节的 CRC-32 值，并将计算结果与服务请求报文中发送的校验值进行比较。
CRC-32 使用如下的生成多项式：
$G(x) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
CRC-32 的初始值使用 IEEE 802.3 规定的 0xFFFFFFFF， 校验结果需要和 0xFFFFFFFF 按位进行异或计算。

4.4.4 依赖性检查

ECU 应该通过验证其软件完整性来检查编程依赖性，包括应用程序与 Bootloader 程序、应用数据与应用程序等。依赖性检查机制由 ECU 供应商制定，并经过知豆电动汽车有限公司的认可和确认。
当 ECU 收到检查编程依赖性例程控制服务的请求时，ECU 将执行依赖性检查,详细信息参考第 5 章。

4.4.5 软件验证

ECU 需定义一个变量，用于标识应用程序是否有效。如果编程完整性检查和编程依赖性检查都正确，那么 ECU 当前程序状态是有效的，标识变量应赋值为 0x5A5A。只有标识变量为 0x5A5A 的应用程序才可以运行，详见第 4.8 节。

4.4.6 Flash 驱动下载

ECU 用于编程的 Flash 驱动并不是存储在 ECU 的非易失性存储器中，而是在编程过程中临时下载到 ECU 的 RAM 中。编程完成后，Flash 驱动将在 ECU 恢复到正常操作模式之前从 RAM 中彻底删除。

具体的 Flash 驱动下载过程，详见第 5 章。

4.4.7 故障容错

ECU 在编程过程中，发生以下情况时，应该可以重编程：
1）从欠压或过压恢复至正常电压；
2）CAN 网络通信故障恢复后；
3）擦除部分 Flash 内存时发生重启。

4.5 源文件格式要求

ECU 供应商释放给知豆电动汽车有限公司的源文件格式是.hex 或.s19。

4.6 通信要求

4.6.1 数据链路层参数

CAN ID 的分配应该遵从知豆电动汽车有限公司的规定，其它的要求详见参考标准[1]。

4.6.2 网络层参数

程序下载的网络层是基于参考标准[3]中的规定实现。
定时参数（如 N_As、N_Bs 等）、BS 和 ST_{min} 将按照参考标准[1]中给定的值来设置。

4.6.3 会话层参数

程序下载的会话层是基于参考标准[4]中的规定实现。
定时参数 S3_Server 与 S3_Client 将按照参考标准[1]中给定的值来设置。

4.6.4 诊断层参数

程序下载的诊断层是基于参考标准[4]中的规定实现。
定时参数（如 P2CAN_Server、P2*CAN_Server 等）将按照参考标准[1]中给定的值来设置。

4.7 诊断服务需求

为了满足 ECU 的编程需求，第 5.3 节定义了应用程序和 Bootloader 程序支持的诊断服务的最小集合。

4.8 ECU 启动时序

在上电/复位后，ECU 执行 Bootloader 程序。Bootloader 程序首先执行一些基本的初始化，然后检查外部编程请求标志位是否置为 TURE。如果外部编程请求标志位置为 TURE，即使应用程序是有效的，Bootloader 程序 也会继续运行。如果当前没有编程请求，则检查应用程序的状态。如果应用程序是有效的(标识变量为 0x5A5A)，则判断在 20ms 内是否收到特定报文。如果收到特定报文，则继续运行 Bootloader 程序；如果没有收到特定报文，则启动应用程序。如果应用程序是无效的(标识变量为 0x0000)，则继续执行 Bootloader 程序。

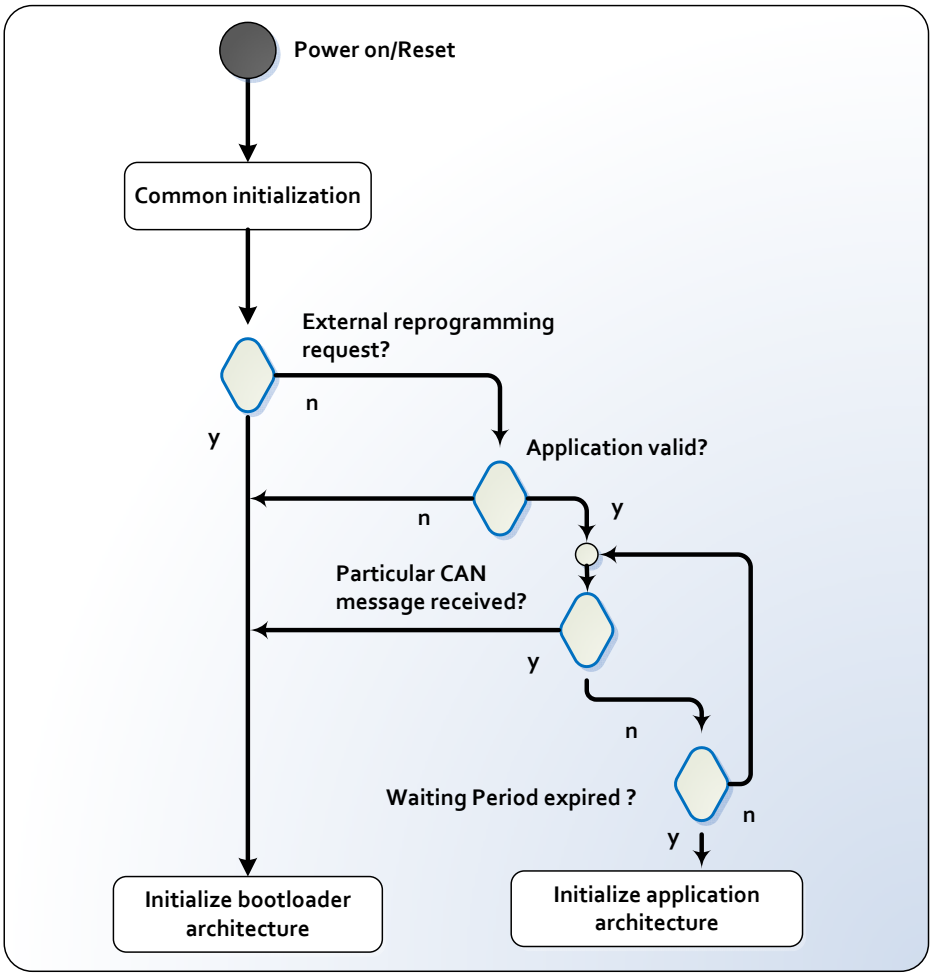

图 1 ECU 启动时序

5 内存编程过程

5.1 Bootloader 启动时序

图 2 描述了 Bootloader 启动时序。在应用模式下，使用了两种不同的诊断会话模式：默认会话和扩展会话。

在 Bootloader 模式下，使用了三种不同的诊断会话模式：默认会话、扩展会话和编程会话。

*注：ECU **不支持**直接从默认会话跳转到编程会话和从编程会话直接跳转到扩展会话。*

在应用模式下，ECU 成功通过预编程阶段后，如果收到“$10 $02”，将置外部编程请求标志位为 TURE，

并执行重启。

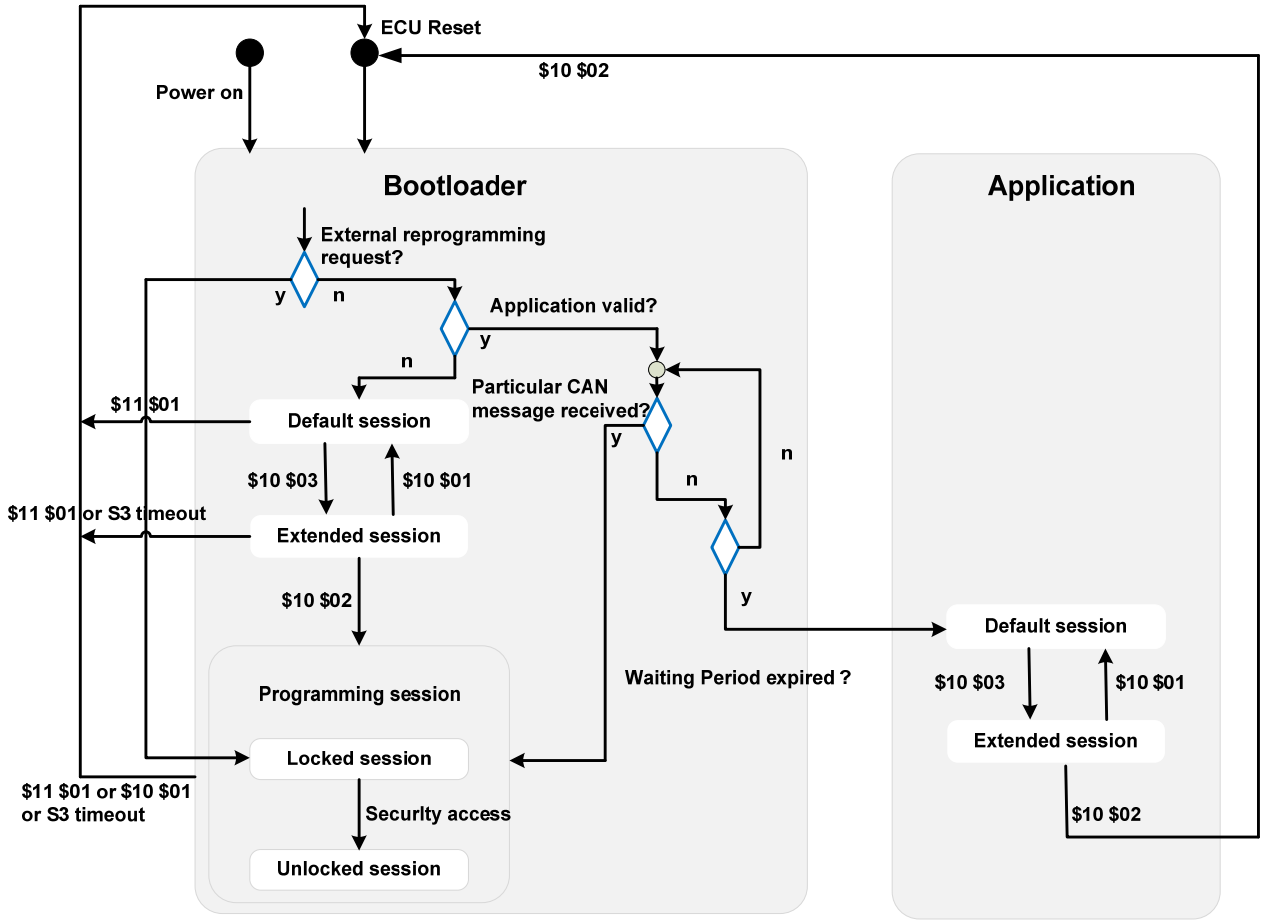

图 2 Bootloader 启动时序

在上电/复位后，ECU 首先执行 Bootloader 引导程序，然后检查外部编程请求标志位。如果外部编程请求标志为 TURE，即使应用程序是有效的，Bootloader 也将继续执行。如果外部编程请求标志为 FASLE，则检查应用程序的状态。

如果应用程序是有效的，则判断在 20ms 内是否收到特定报文。如果收到特定报文，则进入 Bootloader 模式下的编程会话模式；如果没有收到特定报文，则启动应用程序。

特定报文的特征如下：

◆报文的 ID 为 ECU 的**物理寻址**诊断地址；

◆报文的 DLC=8；

◆报文的数据场=04 31 01 02 03 xx xx xx。

注：ECU 在任何诊断会话下收到该特定报文均应给出肯定响应。

如果应用程序是无效的，ECU 停留在 Bootloader 模式下的默认会话模式。在 Bootloader 模式下，诊断会话转换规则与应用模式下相同。

在 Bootloader 模式下，有以下几种方式，可导致 ECU 重启：

◆无论当前处于何种会话模式，“$11 $01”均能重启 ECU。

◆扩展会话模式或编程会话模式下，S3_Server 定时器超时能重启 ECU。

◆在编程会话模式下，“$10 $01”能重启 ECU。

5.2 编程时序

编程时序分为三个编程阶段：

◆预编程阶段：做编程前的网络准备；

◆编程进行阶段：下载程序或数据；

◆后编程阶段：重同步网络。

如果在预编程、编程进行和后编程阶段中，任何物理寻址的请求及响应不满足要求，则全部时序将重新执行，允许重新执行的次数为 1 次。

5.2.1 预编程阶段

编程步骤，如图 3 所示。

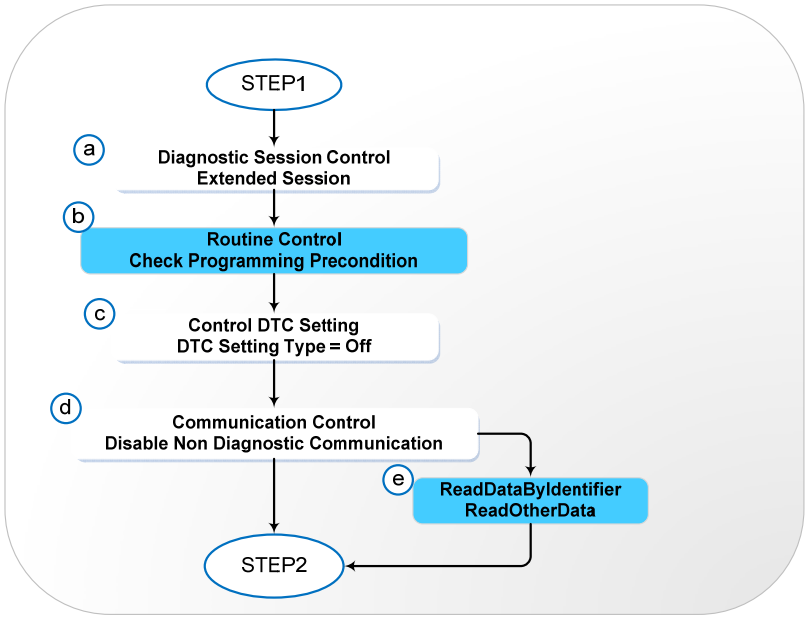

图 3 预编程阶段

（a）诊断会话控制$10 $03：启动扩展会话模式，通过**功能寻址**发送给所有的 ECU。

（b）例程控制“检查预编程条件” $31 $01 $02 $02：通过**物理寻址**检查 ECU 预编程条件，从而确保系统安全，预编程条件由 ECU 决定，如果有任何不安全的因素，ECU 应该拒绝编程，此例程控制不需要安全访问。

注：如果 ECU 在未收到“检查预编程条件”例程($31 $01 $02 $02) 的情况下，收到“$10 $02”请求，ECU 应该拒绝进入 Bootloader 模式，并且发送否定响应。

（c）控制 DTC 设置$85 $02：关闭 DTC 设置，通过**功能寻址**发送给所有的 ECU。

（d）通信控制 0x28 $03 $03：禁止非诊断报文的发送和接收，通过**功能寻址**发送给所有的 ECU。

（e）读取数据 0x22 $xx $yy：在禁止正常通信后，通过**物理寻址**读取预编程 ECU 的状态信息，如：应用软件标识、 应用数据标识、Bootloader 软件标识、VIN 码和指纹记录等。数据读取服务为**可选**服务，读取的内容由 ECU 供应商定义。

5.2.2 编程进行阶段

编程步骤，如图 4 示。所有服务的请求均使用**物理寻址**。

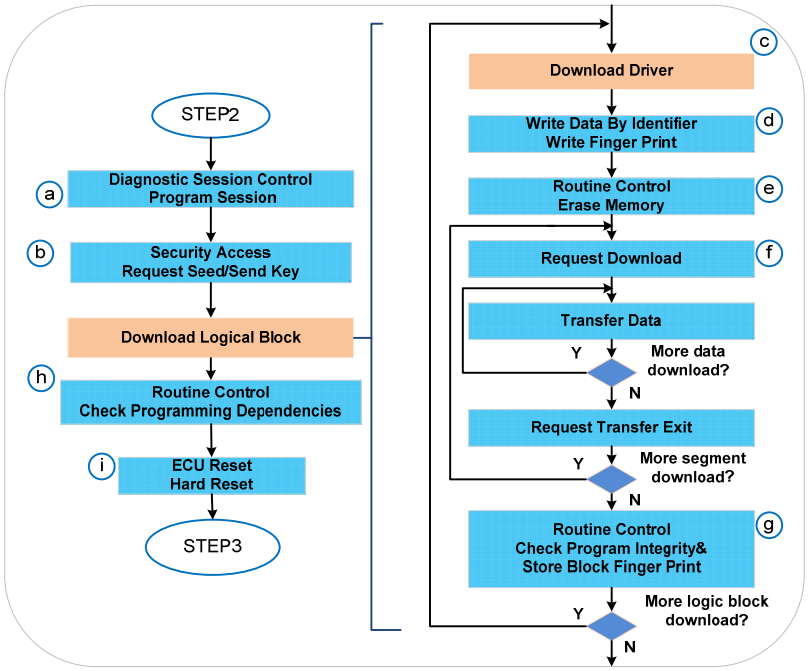

图 4 编程进行阶段

（a）诊断会话控制$10 $02： ECU 收到此请求后，将分配编程所需的资源。ECU 应该在跳转到 Bootloader 模式之前，发送肯定响应。

（b）安全访问$27 $07/$08：编程事件必须通过安全访问，确保只有授权的诊断仪能对 ECU 进行编程操作。

（c）驱动下载$34，$36，$37，$31：Flash 驱动的下载应该按照如下时序来进行：请求下载、传输数据、请求传输退出。下载完所有字节后，用“检查编程完整性”例程($31 $01 $02 $01) 来检查所有的字节是否正确下载。

（d）写入数据$2E $F0 $11：在擦除内存例程（$31 $01 $FF $00）执行之前，ECU 需要将应用数据指纹记录写到内存中。每个逻辑块（除了驱动）下载前，诊断仪都将写一次应用数据指纹记录。当下载完逻辑块后，ECU 根据逻辑块的序号将应用数据指纹记录存储。在追溯指纹记录时，诊断仪将发报文“$22 $F0 $21”，ECU 将发送报文 “$62 $F0 $21…”，根据逻辑块的编号返回每一个逻辑块指纹记录。具体格式，参见 5.3.2 节。

（e）“擦除内存”例程$31 $01 $FF $00：如果擦除内存例程被调用，那么应用程序有效标识变量将被置为无效(0x0000)。

（f）下载过程$34，$36，$37：应用程序或数据的每一个连续的数据块下载到 ECU 非易失性内存中，都需遵循下面的服务顺序完成数下载：

◆请求下载($34)

◆传输数据($36)

◆请求传输退出($37)

（g）“检查编程完整性”例程$31 $01 $02 $01：此例程用来检查所下载的逻辑块的完整性。

（h）“检查编程依赖性”例程$31 $01 $FF $01：完成所有的应用程序或数据的下载，诊断仪将发送检查编程依赖性的例程。检查内容由 ECU 供应商定义，但必须确保所有逻辑块的完整性和一致性。

注：如果整个编程过程只有一个逻辑块，肯定响应的 routineStatusRecord 恒为 00=correctResult，详见

表 13。

（i）电控单元复位$11 $01：诊断仪使用**物理寻址**，发送一个复位类型为硬复位的 ECU 复位服务($11)请求报文到 CAN 网络上。

通过 ECU 复位服务请求将使 ECU 结束编程过程，返回到正常的操作模式。FLASH 驱动程序必须从 RAM 缓存中完全清除，避免非预期的内存擦除。

5.2.3 后编程阶段

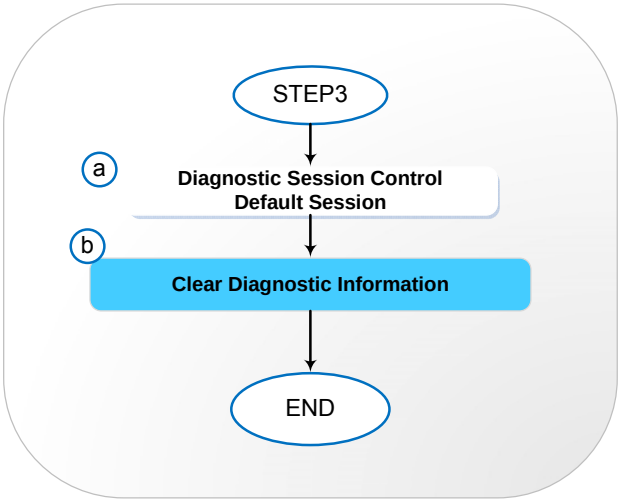

图 5 后编程阶段

（a）诊断会话控制$10 $01：诊断仪通过**功能寻址**发送一个会话类型为默认会话的诊断会话控制$10 服务请求报文到网络上，所有 ECU 进入默认会话模式。

（b）清除诊断信息$14 $FF $FF $FF：诊断仪通过**物理寻址**清除编程 ECU 的诊断信息。

5.3 诊断服务需求

下表定义了编程 ECU 的 Bootloader 程序需要的支持的最小诊断服务集合。为了满足 ECU 的编程，表中的服务必须支持。

表 1 Bootloader 在预编程阶段支持的诊断服务

服务	子功能/数据参数	寻址信息		Bootloader 模式		
		物理	功能	默认	扩展	编程
诊断会话控制$10	会话类型 = 扩诊断会话($03)		√	√	√	
例程控制$31	例程控制类型 = 开始例程 ($01) DID= 检查预编程条件 ($02 $02)	√			√	

通信控制$28	控制类型= 不能接收和发送($03) 通信类型= 应用和网络管理报文 ($03)		√		√	
控制DTC 设置$85	DTC 设置类型= 关 ($02)		√		√	
通过DID 读数据$22	DID = (供应商定义)	√			√	
通过DID 读数据$22	DID = $F0 $21[1]	√				
通过DID 读数据$22	DID = $F0 $11[2]	√				

注:1.此服务在预编程阶段不需要，可能在追溯所有逻辑块的指纹记录时需要。
2.此服务在预编程阶段不需要，可能在追溯最新下载的逻辑块指纹记录时需要。

表 2 Bootloader 在编程进行阶段支持的诊断服务

服务	子功能/数据参数	寻址信息		Bootloader 模式		
		物理	功能	默认	扩展	编程
诊断会话控制($10)	会话类型= 编程诊断会话 ($02)	√			√	√
安全访问($27)	安全访问类型= 请求种子 ($07) 发送密钥($08)	√				√
通过DID 写数据 ($2E)	DID = $F0 $11	√				√
请求下载 ($34)		√				√
数据传输 ($36)		√				√
请求传输退出 ($37)		√				√
例程控制 ($31)	例程控制类型= 开始例程($01) DID = 擦除内存($FF $00)	√				√

例程控制 ($31)	例程控制类型= 开始例程 ($01) DID = 检查编程完整性 ($02 $01)	√				√
例程控制 ($31)	例程控制类型 = 开始例程 ($01) DID = 检查编程依赖性 ($FF $01)	√				√
ECU 复位 ($11)	复位类型= 硬复位 ($01)	√		√	√	√

表 3 Bootloader 在后编程阶段支持的诊断服务

服务	子功能/数据参数	寻址信息		Bootloader 模式		
		物理	功能	默认	扩展	编程
诊断会话控制($10)	会话类型= 默认诊断会话($01)		√	√	√	√
清除诊断信息($14)	$FF $FF $FF	√				

服务的格式参见下节内容，没有在下面列出的服务，参见参考标准[1]。

5.3.1 $2E 服务

Bootloader 中通过数据标识符$F0 $11 写入指纹记录的格式如下表所示。

表 4 写入指纹记录

Byte	Parameter name	Value(hex)
#0	WriteDataByIdentifier Request Service ID	2E
#1	Fingerprint DID （MSB）	F0
#2	Fingerprint DID （LSB）	11
#3	programmingDate Year（BCD-coded）	00-99
#4	programmingDate Month（BCD-coded）	00-12
#5	programmingDate Date（BCD-coded）	00-31
#6	testerSerialNumber(Byte 1, ASCII)	00-FF
#7	testerSerialNumber(Byte 2, ASCII)	00-FF
#8	testerSerialNumber(Byte 3, ASCII)	00-FF

#9	testerSerialNumber(Byte 4, ASCII)	00-FF
…	…	00-FF
#21	testerSerialNumber(Byte 16, ASCII)	00-FF

5.3.2 $22 服务

应用程序中通过标识符$F0 $21 读取指纹记录的格式如下表所示。

表 5 读取指纹记录（读服务肯定响应格式）

Byte	Parameter name	Value(hex)
#0	ReadDataByIdentifier Request Service ID	62
#1	fingerprint DID (MSB)	F0
#2	fingerprint DID (LSB)	21
#3	blockID 0	00-FF
#4	programmingDate Year (BCD-coded)	00-99
#5	programmingDate Month (BCD-coded)	00-12
#6	programmingDate Date (BCD-coded)	00-31
#7~22	testerSerialNumber	00-FF
…	…	…
#n-19	blockID 1	00-FF
#n-18	programmingDate Year (BCD-coded)	00-99
#n-17	programmingDate Month (BCD-coded)	00-12
#n-16	programmingDate Date (BCD-coded)	00-31
# n-15~ n	testerSerialNumber	00-FF

注：ECU 根据下载的地址范围识别出当前下载的是哪个逻辑块。在检查编程完整性例程中，ECU 将$F0$11 对应的指纹记录分别写入相应的逻辑块指纹记录中，诊断仪通过$F0$21 可读取每个逻辑块的指纹记录。

5.3.3 $31 服务

1）检查编程完整性

表 6 检查编程完整性例程请求格式

Byte	Parameter name	Value(hex)
#0	RoutineControl Request Service Id	31
#1	routineControlType= startRoutine	01

#2-3	routineIdentifier= CheckProgrammingIntegrity	0201
#4-7	CRC-32Value，4 bytes	00-FF

表 7 检查编程完整性例程肯定响应格式

Byte	Parameter name	Value(hex)
#0	RoutineControl Request Service Id	71
#1	routineControlType= startRoutine	01
#2-3	routineIdentifier= CheckProgrammingIntegrity	0201
#4	routineStatusRecord= correctResult incorrectResult	 00 01

2）检查预编程条件

表 8 检查预编程条件例程请求格式

Byte	Parameter name	Value(hex)
#0	RoutineControl Request Service Id	31
#1	routineControlType= startRoutine	01
#2-3	routineIdentifier= checkProgrammingPreconditions	0202

表 9 检查预编程条件例程肯定响应格式

Byte	Parameter name	Value(hex)
#0	RoutineControl Request Service Id	71
#1	routineControlType startRoutine	01
#2-3	routineIdentifier checkProgrammingPreconditions	0202
#4	routineStatusRecord PreconditionsOK PreconditionsNotOK	 00 01

3）擦除内存

表 10 擦除内存例程服务请求格式

Byte	Parameter name	Value(hex)
#0	RoutineControl Request Service Id	31
#1	routineControlType startRoutine	01
#2-3	routineIdentifier eraseMemory	FF00
#4	addressAndLength FormatIdentifier lengthFormat（bit 7 – 4）: number of bytes of the memorySize parameter addressFormat（bit 3- 0）:number of bytes of the memoryAddress parameter	0x-4x x0-x4
#5-n1	memoryAddress 1 – 4 bytes erase address	00 - FF
#n2-n3	memorySize 1 – 4 bytes erase size	00 - FF

表 11 擦除内存例程肯定响应格式

Byte	Parameter name	Value(hex)
#0 #1	RoutineControl Request Service Id routineControlType startRoutine	71 01
#2-3	routineIdentifier eraseMemory	FF00
#4	routineStatusRecord correctResult incorrectResult	00 01

4）检查编程依赖性

表 12 检查逻辑块依赖性例程请求格式

Byte	Parameter name	Value(hex)
#0	RoutineControl Request Service Id	31
#1	routineControlType startRoutine	01
#2-3	routineIdentifier checkProgrammingDependencies	FF01

表 13 逻辑块依赖性检查例程肯定响应格式

Byte	Parameter name	Value(hex)
#0	RoutineControl Request Service Id	71
#1	routineControlType startRoutine	01
#2-3	routineIdentifier checkProgrammingDependencies	FF01
#4	routineStatusRecord correctResult incorrectResult	00 01