|  |  |
| --- | --- |
|  | **V1.0**  发布时间： 2019-01 |
| 江苏凌宝新能源车业有限公司 | |
| Bootloader需求规范 | |
|  | |
| 前言： 本文档定义了在指定的江苏凌宝汽车中使用[CAN]总线的电子控制单元[ECU]所遵循的Flash Bootloader重编程行为规范。   |  |  |  |  | | --- | --- | --- | --- | | 编制： | 费博 | 日期： | 2019-01-02 | | 审核： | 黄伟 | 日期： | 2019-01-03 | | 批准： | 黄伟 | 日期： | 2019-01-03 | | |
| 该文档及其所含信息是江苏凌宝的财产。该文档及其所含信息的复制、使用及披露必须得到江苏凌宝的书面授权。 | |

**目 录**

[1 介绍 1](#_Toc516125615)

[1.1 项目描述 1](#_Toc516125616)

[1.2 目标 1](#_Toc516125617)

[1.3 分歧解决 1](#_Toc516125618)

[2 通用需求 2](#_Toc516125619)

[2.1 不可重编程ECU的通用需求 2](#_Toc516125620)

[2.2 重编程ECU的通用需求 2](#_Toc516125621)

[2.2.1 概述 2](#_Toc516125622)

[2.2.2 硬件需求 2](#_Toc516125623)

[2.2.3 软件需求 2](#_Toc516125624)

[2.2.4 安全需求 3](#_Toc516125625)

[2.2.5 源文件格式要求 4](#_Toc516125626)

[2.2.6 通信需求 4](#_Toc516125627)

[2.2.7 诊断服务需求 5](#_Toc516125628)

[2.2.8 ECU启动时序 5](#_Toc516125629)

[2.3 网关ECU需求 5](#_Toc516125630)

[3 重编程流程 6](#_Toc516125631)

[3.1 概述 6](#_Toc516125632)

[3.2 Bootloader启动时序 6](#_Toc516125633)

[3.3 重编程时序 7](#_Toc516125634)

[3.3.1 预编程步骤 7](#_Toc516125635)

[3.3.2 主编程步骤 9](#_Toc516125636)

[3.3.3 后编程步骤 10](#_Toc516125637)

[3.4 诊断服务需求 11](#_Toc516125638)

[3.4.1 通过数据标识符写数据0x2E 13](#_Toc516125639)

[3.4.2 通过标识符读数据0x22 13](#_Toc516125640)

[3.4.3 例程控制——“检查编程完整性”0x31 0x01 0xF0 0x01 14](#_Toc516125641)

[3.4.4 例程控制——“检查编程预条件” 0x31 0x01 0xFF 0x02 15](#_Toc516125642)

[3.4.5 例程控制——“擦除内存”0x31 0x01 0xFF 0x00 15](#_Toc516125643)

[3.4.6 例程控制——“检查编程依赖性” 0x31 0x01 0xFF 0x01 16](#_Toc516125644)

[附 录 A CRC32算法 16](#_Toc516125645)

[附 录 B 文档引用 19](#_Toc516125646)

[附 录 C 术语与缩写 20](#_Toc516125647)

[附 录 D 变更历史 21](#_Toc516125648)

此行切勿删除，目录结束标签

介绍

项目描述

该文档描述了江苏凌宝汽车CAN网络车型平台的Flash Bootloader（重编程）需求。

目标

该文档的目的是ECU供应商阅读并实现其ECU软件。并且，供应商的开发及测试团队也应当明确车辆Flash Bootloader要求，为ECU Flash Bootloader的实现奠定基础。

分歧解决

在当前文档中，遵循以下术语定义：

- “必须shall”表示在文本中的强制性要求；

- “应当should”表示在文本中任选要求；

- “可以can”表示在文本中允许的做法或方法。

如果本标准与其它标准或规范不一致，则按照如下方式处理：

1. 若出现文字描述与配图不符，依照文字描述执行；
2. 与ECU技术规范不一致，遵照ECU技术规范执行；
3. 法规要求高于本技术规范。

通用需求

不可重编程ECU的通用需求

为确保重编程的成功执行，连接在CAN网络上的不可重编程ECU需要满足如下要求：

1. 支持通信控制-禁止/使能发送和接收（即0x28服务中的0x00/0x03子功能）的诊断服务，通过此服务，可以降低编程过程中总线的负载；
2. 支持控制[DTC]设置（0x85）诊断服务，通过此服务，确保当某个ECU处于编程状态时，其他ECU不会设置DTC；
3. 不支持重编程相关的诊断服务，如：诊断会话控制-切换到编程模式（0x10 0x02）、请求下载（0x34）、传输数据（0x36）、请求传输退出（0x37）。

不可重编程ECU需要支持第3.3.1节中规定的诊断服务，除了“例程控制-检查预编程条件”以及“通过标识符读数据”两个服务。

重编程ECU的通用需求

概述

所有支持应用软件及应用数据（包括网络配置数据和标定数据）重编程的ECU，应当包含Bootloader软件。在正常运行过程中，执行的是应用软件和应用数据。仅当应用软件或应用数据无效时，或者要求对此进行升级的时候，Bootloader软件才被激活。

应用软件和应用数据可以同时编程或者相互独立编程，不允许重编程时更新Bootloader软件。

硬件需求

任何可重编程的并且正确安装在汽车中的ECU可通过[VPDIAG]中规定的诊断接口进行编程。在重编程过程中不需要把ECU从整车中移除。

可重编程的ECU必须提供足够的存储器来保证下载，提供充足的缓冲空间来满足重编程定时要求。用于重编程的网络层缓冲区所占的字节数应该得到江苏凌宝汽车认可。

可重编程（擦除/写入）的次数将在[CTS]中规定，并应该得到江苏凌宝汽车认可。

软件需求

Bootloader软件必须存储在被保护的存储器中，从而确保即使存在潜在错误，ECU始终是可重编程的。

（可选）当ECU初始化完成且停留在Bootloader模式下，则ECU Bootloader将启动一个睡眠定时器，如果收到CAN诊断报文则重启睡眠定时器，当睡眠定时器超时后，则Bootloader将ECU置为低功耗状态，推荐I/O端口设置成低功耗状态，且ECU在Bootloader模式下需支持网络唤醒。推荐睡眠定时器为300s。

安全需求

为确保下载的安全性，ECU应该实现一些安全需求，避免下面几种情况：

1. 来自非法源的下载动作；
2. 当前重编程条件不满足；
3. 下载错误的应用软件或者应用数据到ECU；
4. 软件之间不兼容。

安全访问

所有可重编程的ECU应该支持种子和密钥的安全特性，并且可以通过安全访问服务（0x27）进入访问，从而保护ECU免遭未授权的编程动作影响。

安全等级0x03/0x04用于编程会话模式，详细的要求请参考[VPDIAG]。

编程预条件

ECU应该确保重编程的执行是处于安全状态。如果编程预条件不满足（如车辆或发动机正在运行等），那么重编程请求将被拒绝。

通过一个“检查编程预条件”例程控制来激活ECU编程预条件的检验，详细信息请参考第3章。

完整性验证

ECU需要检查下载到存储器中的数据的完整性。当一个逻辑块下载后，将使用CRC32算法验证当前逻辑块的所有数据字节是否被正确传输和写入。通过一个“检查编程完整性”例程控制来激活ECU完整性验证。当ECU接收到此服务请求时，Bootloader将计算下载数据字节的CRC32值，并将计算结果与诊断仪请求报文中发送的校验值进行比较。

CRC32使用如下的多项式生成：

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

IEEE 802.3规定CRC-32使用的初始值为0xFFFFFFFF。

CRC32算法详见“附录A”,附录A中提供两种CRC32算法。

依赖性检查

不兼容的软件不能配合使用，如果配合使用可能会使功能异常或产生致命性错误。因此，ECU应该通过验证软件兼容性来检查重编程依赖性，包括应用软件与Bootloader软件、应用数据与应用软件等。依赖性检查机制由ECU供应商制定，并经得江苏凌宝汽车诊断工程师批准。

当ECU收到例程控制—“检查编程依赖性”诊断服务的请求时，ECU将执行依赖性检查。详细的定义请参考第3章。

软件有效性验证

ECU内部定义一个标志位，用于标识应用软件是否有效。如果重编程完整性检查和重编程依赖性检查都正确，ECU才设置应用软件的标志位为有效。只有标志位为有效时，应用软件才可以运行（详见第2.2.8节）。

内存驱动下载

内存驱动是一种执行初始化、擦除或写入内存功能的硬件从属软件。

ECU内存的内容必须是受保护的，防止意外的擦除或重写操作。因此，ECU需要采用软件互锁机制，就是说用于编程的内存驱动并不是存储在ECU的永久性存储器中，而是在下载过程中下载到ECU的RAM中。下载完成后，驱动代码将从ECU RAM缓冲区彻底移除。

具体的内存驱动下载过程，详见第3章。

故障容错

Bootloader软件将被存储在一个受保护的永久性存储器中，这样可以避免被意外擦除，即使没有应用软件或应用数据，ECU依然可以支持重编程。

ECU在重编程过程中，发生以下情况时，应该可以重编程：

1. 从低/高电压状态恢复；
2. CAN通信错误和超时恢复后；
3. 重启；
4. 擦除了部分Flash内存。

源文件格式要求

发布给江苏凌宝汽车的源文件格式是Motorola格式（s19）。

通信需求

数据链路层参数

CAN ID的分配应该遵从江苏凌宝汽车的规定，而且应该与增强型诊断中用的CAN ID一致。

其他的要求详见[VPDIAG]。

网络层参数

内存下载的网络层实现是基于[ISO2]中的规定。

定时参数（如N\_As、N\_Bs等）、BS和STmin将按照[VPDIAG]中给定的值来设置。

诊断层参数

内存下载的诊断层需遵循[ISO3]中的值来设置。

定时参数（如P2server、S3server等）需遵循[VPDIAG]中给定的值来设置。

诊断服务需求

为了满足存储器编程的执行需求，第3.4节定义了应用软件和Bootloader软件支持的诊断服务的最小设置。

ECU启动时序

在上电/复位后，ECU首先执行Bootloader代码。Bootloader首先执行一些基本的初始化，然后检查外部重编程请求标志位是否为有效，如果重编程请求标志位为有效，即使应用程序是有效的，Bootloader也会继续运行。如果当前没有重编程请求，即重编程请求标志位为无效，则检查应用软件的状态：如果应用软件是有效的，则应用软件代码将被执行；如果应用软件是无效的，则继续执行Bootloader代码。



1. ECU启动时序

网关ECU需求

如果诊断报文需要通过网关转发，需要遵循下面规定：

1. 地址为网关本身的物理寻址的诊断请求报文，则此诊断请求报文不需要被路由；
2. 地址为非网关本身的物理寻址的诊断请求报文，则此诊断请求报文需要被路由到相应的网段；
3. 地址为功能寻址的诊断请求报文，则网关必须将其路由到其连接的各个网段。

如果对网关ECU本身进行编程，则与非网关ECU不同的是：网关ECU的Bootloader必须在其连接的网段上路由或者发出诊断仪在线报文（0x3E）。

重编程流程

概述

本章定义了将一个或多个应用软件或应用数据下载到ECU内存中的重编程流程。

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：默认会话模式和扩展会话模式。

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

从诊断会话的角度来说，要进入编程会话模式，必须先通过扩展会话模式。这意味着ECU不支持直接从默认会话直接跳转到编程会话模式。同样地，ECU也不支持从编程会话直接跳转到扩展会话模式。

如果ECU在正确的条件下收到“0x10 0x02”，ECU将外部重编程请求标志位置为有效，并执行ECU重启。



1. Bootloader启动时序

上电/复位后，ECU首先执行Bootloader引导代码，然后检查外部重编程请求标志位：

1. 如果外部重编程请求标志位为有效，那么即使应用程序是有效的，Bootloader也会继续进一步执行，在此情况下，ECU直接进入编程会话模式。
2. 如果外部重编程请求标志位为无效，则继续检查应用软件的标志位状态：
3. 如果应用软件是有效的，则启动应用模式；
4. 如果应用软件无效，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，会导致ECU重启：

1. 无论当前处于何种会话模式，“0x11 0x01”都会引导ECU重启；
2. 在扩展会话模式或编程会话模式下，S3定时器超时会导致ECU重启；
3. 在编程会话模式下，“0x10 0x01”会导致ECU重启。

重编程时序

重编程时序分为三个编程步骤：

1. 预编程步骤：编程前的CAN网络准备；
2. 主编程步骤：下载应用软件或应用数据；
3. 后编程步骤：重同步CAN网络。

后面的章节详细描述了这三个步骤。在图3、图4及图5中的描述中，可选步骤放在了序列的右边，蓝色框和橙色框代表物理寻址，白色框代表功能寻址。

如果在预编程、主编程和后编程步骤过程中，任何物理寻址的请求及响应不满足要求，则全部时序须被再次执行，但诊断仪最多允许重新执行一次。

预编程步骤

预编程步骤用来为要下载的ECU做重编程前的CAN网络准备。此步骤也包含了提高下载速度的准备。此步骤的请求报文能采用的是物理寻址，或功能寻址。预编程步骤，如图3所示。



1. 预编程步骤

（a）诊断会话控制0x10 0x03：为了禁止ECU间的正常通信和控制DTC设置，预编程需要启动非默认会话模式。通过使用会话类型为扩展会话模式的诊断会话控制（0x10）服务来完成。此请求使用一个单帧请求报文，通过功能寻址发送给所有的ECU。

（b）例程控制“检查编程预条件”0x31 0x01 0xFF 0x02：通过此例程来检查ECU编程条件，从而确保系统安全，预编程检验条件由ECU决定，如果有任何不安全的因素，ECU应该拒绝编程。

注意：如果ECU在未收到“检查编程预条件”例程（0x31 0x01 0xFF 0x02）的情况下，收到“0x10 0x02”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0x85 0x02：诊断仪通过DTC设置类型设为“关闭”的控制DTC设置服务请求。此请求使用一个单帧请求报文，通过功能寻址发送给所有的ECU。

（d）通信控制0x28 0x03 0x03：诊断仪通过通信控制（0x28）服务请求，禁止非诊断报文的发送和接收。请求中的控制类型参数置为“disable the transmission and the reception”，通信类型置为“application and network management messages”。此请求使用一个单帧请求报文，通过功能寻址发送给所有的ECU。

（e）读取数据0x22 0xxx 0xyy：在禁止正常通信后，读取被编程的ECU的状态（如：编程的应用软件和数据）。从被编程的ECU读取服务器标识数据，标识如应用软件标识、应用数据标识、Bootloader软件标识。读取数据服务为可选服务，读取的内容由ECU供应商定义。

主编程步骤

在预编程步骤之后，是主编程步骤。主编程时序是单个ECU编程事件的应用，因此所有服务的请求都使用物理寻址。

图4描述了在主编程步骤中执行的各个服务。



**Write Data By Identifier**

**Write Finger Print(Optional)**

**Download Driver(Optional)**

1. 主编程步骤

（a）诊断会话控制0x10 0x02：在收到一个寻址方式为物理寻址，子功能为编程会话的诊断会话控制（0x10）服务后，ECU启动Bootloader，并分配编程所需的所有资源。ECU 跳转到编程模式之后再发送肯定响应。

（b）安全访问0x27 0x03/0x04：编程事件必须通过安全访问。安全访问（0x27）服务在排放相关和安全系统中是强制的。其它系统不要求使用该服务。下载前，通过安全访问过程是强制的，确保只有合法的诊断仪能对ECU进行下载操作。

为了缩短重编程时间，上电可直接执行安全访问服务，详细描述参照[VPDIAG]中的安全访问机制。

（c）驱动下载(可选项)0x34，0x36，0x37，0x31：当ECU的非易失性存储单元中没有存储内存驱动时，将执行内存驱动的下载，否则跳过此步骤。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（0x31 0x01 0xF0 0x01）来检查所有的字节都正确传输。

（d）写入数据(可选项)0x2E 0xF1 5A：在擦除内存例程之前，将“指纹”写到ECU内存中是非强制的。“指纹”标识了是哪个诊断仪对ECU内存做了修改。我们使用0xF1 5A数据标识符而不是引导软件指纹、应用软件指纹、应用数据指纹这些数据标识符。每个逻辑块（除了驱动）下载前，诊断仪将首先写“指纹”，在下载完逻辑块后，ECU即知道之前下载的程序是哪个逻辑块（即逻辑块序号），并根据逻辑块的序号将它存储。在追踪指纹信息时，诊断仪将发报文“0x22 0xF1 5B”，ECU将通过“0x62 0xF15B…”，返回每一个逻辑块的指纹信息。具体格式，请参看章节3.4.2。

（e）例程控制——“擦除内存”0x31 0x01 0xFF 0x00：为了允许应用软件和数据下载，ECU的内存将被擦除。此步骤通过例程控制服务（0x31）来执行擦除内存。如果擦除内存例程被调用执行，那么应用软件的标志位将被置为无效。

（f）下载过程0x34，0x36，0x37：应用软件或者数据的每一个连续的数据块（也叫段，可能是一个完整的应用软件或者数据，也可能是应用软件或者数据的一部分）下载到ECU非易失性内存中，都是遵循下面的服务顺序完成数传输：

1. 请求下载（0x34）；
2. 传输数据（0x36）；
3. 请求传输退出（0x37）。

单个应用软件或数据块可能需要多个数据传输（0x36）请求报文来完成传输（当数据块长度超出网络层缓存大小时，就会出现这种情况）。

（g）例程控制——“检查编程完整性”0x31 0x01 0xF0 0x01：此例程用来检查逻辑块的完整性。

（h）例程控制——“检查编程依赖性”0x31 0x01 0xFF 0x01：一旦完成所有的应用软件或数据块/模块的下载，诊断仪将开始一个例程来触发ECU检查重编程的依赖性。ECU供应商定义检查内容，但必须确保所有逻辑块的兼容性和一致性。

（i）电控单元复位0x11 0x01：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（0x11）服务请求报文到CAN网络上。

通过ECU复位服务请求将使ECU结束重编程过程，返回到正常的操作模式。内存驱动代码必须从RAM缓存中完全清除，避免意外激活这些可能会进行非预期的内存擦除或程序操作的代码。

后编程步骤



1. 后编程步骤

（a）诊断会话控制0x10 0x01：诊断仪发送一个会话类型为默认会话的诊断会话控制（0x10）服务请求报文到CAN网络上。所有的ECU接收到诊断会话控制（0x10），而进入到默认会话模式。此请求通过功能寻址发送，请求发送给所有包含在编程事件中或因此而进入非默认会话模式的ECU。跳转到默认会话模式，表示通信控制（0x28）服务和控制DTC设置（0x85）服务也将复位到默认状态。

（b）清除诊断信息0x14 0xFF 0xFF 0xFF：如果重编程ECU在编程步骤被重启，当编程ECU运行在默认会话模式时，网络上其它的ECU仍然在不能正常通信状态，此时，一些可能被存储在编程ECU中的诊断信息就应该通过物理寻址的清除诊断信息（0x14）服务来清除。

诊断服务需求

下表定义了一个重编程ECU的Bootloader软件需要的最小诊断服务。为了满足执行ECU非易失性内存的编程，表格中的服务必须要支持。

1. 预编程步骤支持的诊断服务

| **服务** | **子功能/数据参数** | **寻址信息** | | **应用模式** | | **Bootloader 模式** | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **物理** | **功能** | **默认** | **扩展** | **默认** | **扩展** | **编程** |
| 诊断会话控制 （0x10） | 会话类型 = 扩展诊断会话 （0x03） |  | √ | √ | √ | √ | √ |  |
| 例程控制 （0x31） | 例程控制类型 = 开始例程 （0x01）  DID = 检查编程预条件 （0xFF 0x02） | √ |  |  | √ |  | √ |  |
| 通信控制 （0x28） | 控制类型 = 不能接收和发送 （0x03）  通信类型 = 应用和网络管理报文 （0x03） |  | √ |  | √ |  | √ |  |
| 控制DTC设置 （0x85） | DTC设置类型 = 关 （0x02） |  | √ |  | √ |  | √ |  |
| 通过DID读数据  （22h） | DID = 0x F15A | √ |  | √ | √ | √ | √ |  |
| 通过DID读数据  （22h） | DID = 0x F15B | √ |  | √ | √ | √ | √ |  |

注意:

√: 需要支持的服务

1: 此服务在前编程步骤不需要，但可能在追踪所有逻辑块的“指纹”时需要。

2: 此服务在前编程步骤不需要，但可能在追踪最新下载的逻辑块“指纹”时需要。

1. 主编程步骤支持的诊断服务

| **服务** | **子功能/数据参数** | **寻址信息** | | **应用模式** | | **Bootloader 模式** | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **物理** | **功能** | **默认** | **扩展** | **默认** | **扩展** | **编程** |
| 诊断会话控制（0x10） | 会话类型 = 编程诊断会话（0x02） | √ |  |  | √ |  | √ | √ |
| 安全访问  （0x27） | 安全访问类型 = 请求种子（0x03）  发送密钥（0x04） | √ |  |  |  |  |  | √ |
| 通过DID写数据（0x2E） | DID = 0xF1 0x5A | √ |  |  |  |  |  | √ |
| 请求下载（0x34） |  | √ |  |  |  |  |  | √ |
| 数据传输（0x36） |  | √ |  |  |  |  |  | √ |
| 请求传输退出（0x37） |  | √ |  |  |  |  |  | √ |
| 例程控制（0x31） | 例程控制类型 = 开始例程（0x01）  DID = 擦除内存（0xFF 0x00） | √ |  |  |  |  |  | √ |
| 例程控制（0x31） | 例程控制类型 = 开始例程（0x01）  DID = 检查编程完整性（0xF0 0x01） | √ |  |  |  |  |  | √ |
| 例程控制（0x31） | 例程控制类型 = 开始例程（0x01）  DID = 检查编程依赖性（0xFF 0x01） | √ |  |  |  |  |  | √ |
| ECU复位（0x11） | 复位类型 = 硬复位（0x01） | √ |  |  |  | √ | √ | √ |

1. 后编程步骤支持的诊断服务

| **服务** | **子功能/数据参数** | **寻址信息** | | **应用模式** | | **Bootloader 模式** | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **物理** | **功能** | **默认** | **扩展** | **默认** | **扩展** | **编程** |
| 诊断会话控制（0x10） | 会话类型 =  默认诊断会话 （0x01） |  | √ |  | √ | √ | √ | √ |
| 清除诊断信息 （0x14） |  | √ |  | √ |  |  |  |  |

服务的格式请参看下节内容，没有在下面列出的服务，请参考[VPDIAG]。

通过数据标识符写数据0x2E

Bootloader中的通过数据标识符写数据服务支持数据标识符 0xF15A—“指纹”，此数据标识符应该是可读可写的（R/W）。

1. 写入数据-“指纹”

| **Byte** | **Parameter name** | **Value（hex）** |
| --- | --- | --- |
| #0 | WriteDataByIdentifier Request Service ID | 2E |
| #1 | Fingerprint DID （HByte） | F1 |
| #2 | Fingerprint DID （LByte） | 5A |
| #3 | programmingDate YY （byte 0, BCD-coded） | 00-99 |
| #4 | programmingDate MM （byte1, BCD-coded） | 01-12 |
| #5 | programmingDate DD （byte 2, BCD-coded） | 01-31 |
| #6~#11 | testerSerialNumber | 00-FF |

通过标识符读数据0x22

应用程序中的通过标识符读数据服务支持数据标识符0xF15B—“指纹”，通过此服务可以追踪ECU每个逻辑块的指纹信息，其遵循表5所示的格式：

1. 读“指纹”数据服务肯定响应格式

| **Byte** | **Parameter name** | **Value（hex）** |
| --- | --- | --- |
| #0 | ReadDataByIdentifier Request Service ID | 62 |
| #1 | fingerprint DID（HByte） | F1 |
| #2 | fingerprint DID（LByte） | 5B |
| #3 | blockID 0 | 00-FF |
| #4 | programmingDate Year （HByte,BCD-coded） | 00-99 |
| #5 | programmingDate Month （MByte,BCD-coded） | 01-12 |
| #6 | programmingDate Date （LByte, BCD-coded） | 01-31 |
| #7-12 | testerSerialNumber | 00-FF |
| #13 | blockID 1 | 00-FF |
| #14 | programmingDate Year （HByte,BCD-coded） | 00-99 |
| #15 | programmingDate Month （MByte,BCD-coded） | 01-12 |
| #16 | programmingDate Date （LByte, BCD-coded） | 01-31 |
| #17-22 | testerSerialNumber | 00-FF |
| #23 | … | … |
| #2+N\*10 | testerSerialNumber | 00-FF |
| #3+N\*10 | blockID N-1 | 00-FF |
| … | … | … |
| 注：N -逻辑块的数量 | | |

为实现诊断仪追踪所有逻辑块下载的指纹信息，因此ECU需要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需将0xF15A中存储的指纹信息放在对应的逻辑块指纹信息中，诊断仪可通过0xF15B读出每个逻辑块的指纹信息。

例程控制——“检查编程完整性”0x31 0x01 0xF0 0x01

1. 例程控制——“检查编程完整性”请求格式

| **Byte** | **Parameter name** | **Value（hex）** |
| --- | --- | --- |
| #0 | RoutineControl Request Service Id | 31 |
| #1 | routineControlType  startRoutine | 01 |
| #2-3 | routineIdentifier  CheckProgrammingIntegrity | F001 |
| #4-7 | routineControlOptionRecord1 | 00-FF |
| 注：1 此四个字节是诊断仪Checksum计算的结果 | | |

1. 例程控制——“检查编程完整性”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 | F001 |
| #4 | routineStatusRecord  correctResult  incorrectResult | 00  01 |

例程控制——“检查编程预条件” 0x31 0x01 0xFF 0x02

1. 例程控制——“检查编程预条件”请求格式

| **Byte** | **Parameter name** | **Value（hex）** |
| --- | --- | --- |
| #0 | RoutineControl Request Service Id | 31 |
| #1 | routineControlType  startRoutine | 01 |
| #2-3 | routineIdentifier  checkProgrammingPreconditions | FF02 |

1. 例程控制——“检查编程预条件”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 | FF02 |
| #4 | routineStatusRecord  correctResult  incorrectResult | 00  01 |

例程控制——“擦除内存”0x31 0x01 0xFF 0x00

1. 例程控制——“擦除内存”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 |

1. 例程控制——“擦除内存”肯定响应格式

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

例程控制——“检查编程依赖性” 0x31 0x01 0xFF 0x01

1. 例程控制——“逻辑块依赖性检查”请求格式

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

1. 例程控制——“逻辑块依赖性检查”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 |

1. CRC32算法

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*算法一：通过查表法进行运算，这种方法运算速度快，但代码长度长\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static unsigned \_\_int32 crctab[256] = {

0x00000000,0x04c11db7,0x09823b6e,0x0d4326d9, 0x130476dc, 0x17c56b6b,

0x1a864db2, 0x1e475005, 0x2608edb8, 0x22c9f00f, 0x2f8ad6d6, 0x2b4bcb61,

0x350c9b64,0x31cd86d3, 0x3c8ea00a, 0x384fbdbd, 0x4c11db70, 0x48d0c6c7,

0x4593e01e,0x4152fda9, 0x5f15adac, 0x5bd4b01b, 0x569796c2, 0x52568b75,

0x6a1936c8,0x6ed82b7f,0x639b0da6, 0x675a1011, 0x791d4014, 0x7ddc5da3,

0x709f7b7a,0x745e66cd,0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039,

0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5, 0xbe2b5b58, 0xbaea46ef,

0xb7a96036,0xb3687d81,0xad2f2d84, 0xa9ee3033, 0xa4ad16ea, 0xa06c0b5d,

0xd4326d90, 0xd0f37027, 0xddb056fe, 0xd9714b49, 0xc7361b4c, 0xc3f706fb,

0xceb42022, 0xca753d95, 0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1,

0xe13ef6f4, 0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d, 0x34867077, 0x30476dc0,

0x3d044b19,0x39c556ae,0x278206ab,0x23431b1c, 0x2e003dc5, 0x2ac12072,

0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcdbb16, 0x018aeb13, 0x054bf6a4,

0x0808d07d,0x0cc9cdca,0x7897ab07, 0x7c56b6b0, 0x71159069, 0x75d48dde,

0x6b93dddb, 0x6f52c06c, 0x6211e6b5, 0x66d0fb02, 0x5e9f46bf, 0x5a5e5b08,

0x571d7dd1,0x53dc6066,0x4d9b3063,0x495a2dd4,0x44190b0d, 0x40d816ba,

0xaca5c697, 0xa864db20, 0xa527fdf9, 0xa1e6e04e, 0xbfa1b04b, 0xbb60adfc,

0xb6238b25, 0xb2e29692, 0x8aad2b2f, 0x8e6c3698, 0x832f1041, 0x87ee0df6,

0x99a95df3,0x9d684044,0x902b669d,0x94ea7b2a, 0xe0b41de7, 0xe4750050,

0xe9362689, 0xedf73b3e, 0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2,

0xc6bcf05f, 0xc27dede8, 0xcf3ecb31, 0xcbffd686, 0xd5b88683, 0xd1799b34,

0xdc3abded, 0xd8fba05a, 0x690ce0ee, 0x6dcdfd59, 0x608edb80, 0x644fc637,

0x7a089632,0x7ec98b85,0x738aad5c, 0x774bb0eb, 0x4f040d56, 0x4bc510e1,

0x46863638,0x42472b8f,0x5c007b8a, 0x58c1663d, 0x558240e4, 0x51435d53,

0x251d3b9e, 0x21dc2629, 0x2c9f00f0, 0x285e1d47, 0x36194d42, 0x32d850f5,

0x3f9b762c, 0x3b5a6b9b, 0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff,

0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623, 0xf12f560e, 0xf5ee4bb9,

0xf8ad6d60, 0xfc6c70d7, 0xe22b20d2, 0xe6ea3d65, 0xeba91bbc, 0xef68060b,

0xd727bbb6,0xd3e6a601,0xdea580d8, 0xda649d6f, 0xc423cd6a, 0xc0e2d0dd,

0xcda1f604, 0xc960ebb3, 0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7,

0xae3afba2, 0xaafbe615, 0xa7b8c0cc, 0xa379dd7b, 0x9b3660c6, 0x9ff77d71,

0x92b45ba8,0x9675461f, 0x8832161a, 0x8cf30bad, 0x81b02d74, 0x857130c3,

0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640, 0x4e8ee645, 0x4a4ffbf2,

0x470cdd2b, 0x43cdc09c, 0x7b827d21, 0x7f436096, 0x7200464f, 0x76c15bf8,

0x68860bfd,0x6c47164a,0x61043093, 0x65c52d24, 0x119b4be9, 0x155a565e,

0x18197087, 0x1cd86d30, 0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,

0x3793a651,0x3352bbe6,0x3e119d3f, 0x3ad08088, 0x2497d08d, 0x2056cd3a,

0x2d15ebe3,0x29d4f654, 0xc5a92679, 0xc1683bce, 0xcc2b1d17, 0xc8ea00a0,

0xd6ad50a5, 0xd26c4d12, 0xdf2f6bcb, 0xdbee767c, 0xe3a1cbc1, 0xe760d676,

0xea23f0af, 0xeee2ed18, 0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4,

0x89b8fd09,0x8d79e0be, 0x803ac667, 0x84fbdbd0, 0x9abc8bd5, 0x9e7d9662,

0x933eb0bb, 0x97ffad0c, 0xafb010b1, 0xab710d06, 0xa6322bdf, 0xa2f33668,

0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4

};

unsigned int CalcCRCt(int size, unsigned char \*data)

{

unsigned \_\_int32 crc = 0xffffffff; /\* initial value \*/

int i;

for (i = 0; i < size; i++)

{

crc = (crc << 8) ^ crctab[((crc >> 24) ^ data[i])&0xff];

}

return crc;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 算法二：直接运算数据，这种方法运算时间长，但代码长度短 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned int CalcCRC(int size, unsigned char \*data)

{

unsigned \_\_int32 crc = 0xffffffff; /\* initial value \*/

int i, j;

for (i = 0; i < size; i++) {

crc = crc ^ (data[i] << 24);

for (j = 0; j < 8; j++) {

if (crc & 0x80000000)

crc = (crc << 1) ^ 0x04c11db7;

else

crc <<= 1;

}

}

return crc;

}

1. 文档引用

|  |  |  |
| --- | --- | --- |
| **编号** | **日期** | **描述** |
| VPDIAG | 2017 | 江苏凌宝汽车CAN网络开发\_UDS诊断需求规范.docx |
| ISO1 | 2013 | ISO 14229-1: Road vehicles - Unified diagnostic services （UDS） - Part 1:Specification and requirements |
| ISO2 | 2011 | ISO/DIS 15765-2: Road vehicles - Diagnostics on CAN - Part 2: Network layer services |
| ISO3 | 2006 | ISO/DIS 15765-3: Road vehicles - Diagnostics on CAN - Part 3: Implementation of Diagnostic Services |

1. 术语与缩写

|  |  |
| --- | --- |
| **名称** | **描述** |
| ECU | Electric Control Unit电子控制单元 |
| DTC | Diagnostic Trouble Code 故障诊断代码 |
| CTS | Component Technical Specification部件技术规范 |

1. 变更历史

|  |  |  |
| --- | --- | --- |
| **版本** | **更改日期** | **更改描述** |
| 1.0 | 2019-01-02 | 费博：初始版本 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |