文件编号：WTK-NV1x-CC001-00DQ04

版 本 号：V1.1

密 级：内部

**NV1x项目EDR读取设备**

**固件设计报告**

**编制部门： 研发中心**

**编 制：**

**审 核：**

**批 准：**

**会 签：**

**归 档：**

南 京 维 思 科 汽 车 科 技 有 限 公 司

版本修正记录：

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **更改日期** | **更改内容** | **更改人** |
| V1.0 | 2022-05-20 | 初次编制 | 张宇 |
| V1.1 | 2022-05-27 | 更新6.2.2章节UART通讯交互协议内容(下位机反馈) | 张宇 |
|  |  |  |  |

目录

**[1. 概述 3](#_Toc18184)**

[1.1. 编写目的 3](#_Toc28564)

[1.2. 项目背景 3](#_Toc25478)

[1.3. 参考资料 3](#_Toc9242)

**[2. 需求分析 4](#_Toc7571)**

[2.1. 固件驱动层需求分析 4](#_Toc3643)

[2.2. 固件应用层需求分析 4](#_Toc14582)

**[3. 系统概念设计 5](#_Toc18140)**

**[4. 软件架构设计 6](#_Toc9818)**

[4.1. 硬件驱动层概述 6](#_Toc4614)

[4.2. 系统服务层概述 7](#_Toc32635)

[4.3. 应用任务层概述 9](#_Toc30941)

**[5. 驱动层模块设计 9](#_Toc3562)**

[5.1. External I/O - 外部输出输入控制信号驱动 9](#_Toc14296)

[5.2. CAN2.0 A/B通信驱动 9](#_Toc30910)

[5.3. UART通信驱动 10](#_Toc19502)

[5.4. FTM灵活定时器驱动 10](#_Toc22756)

**[6. 应用层模块设计 11](#_Toc355)**

[6.1. UDSonCAN模块应用设计 11](#_Toc12532)

[6.2. UART通讯模块应用设计 24](#_Toc10308)

[6.3. 设备加密模块设计 29](#_Toc27832)

**[7. 方案总结 30](#_Toc10978)**

# 概述

## 编写目的

针对NV1x项目EDR读取设备的系统需求，从系统功能特征的识别，到系统固件的模块(驱动层、系统层、应用层)设计，完整描述系统固件的实现细节，以形成系统固件代码开发测试的输入文件，同时作为EDR读取设备开发的评审文件之一。

## 项目背景

当前ACU或EDR产品在整车端试验时，需要携带CAN分析仪、笔记本电脑等固化的模块，造成了便捷性的约束，并且由于设备的加密性，对外使用非常不便，并且操作繁琐，且有发生意外的风险，因此需要一个标准设备，简化并统一流程，将人为操作的风险降低。

## 参考资料

1. NV1x项目EDR读取设备需求与开发方案
2. EDR读取设备设计说明书-V0.0-20190321.DOCX
3. 《S32K-RM.pdf》
4. ACU固件OSEK系统概述与设计分析
5. NV1x-CC001项目UDSonCAN模块软件设计
6. 串口总线（485）通信协议规范v1.5.DOCX

# 需求分析

根据“NV1x项目EDR读取设备需求与开发方案”的描述，在EDR读取设备的使用特性与功能特性方面，根据功能定义、使用场景、以及考虑到使用的广泛性与便捷性，相应的需求分为如下几类：

* 设备的使用场景为配置OBD端口的整车环境，能够建立CAN网络整车通讯读取UDS服务；
* 设备能够与上位机建立通讯，将整车端数据反馈至人机交互端口；
* 设备正常工作或通讯正常时需要点亮信号灯提示；
* 设备的加密属性，即仅能针对项目车辆进行使用，而不能随意建立通讯；

针对以上四个概述，分别对“在哪里”、“做什么”、“怎么做”、“结果展示”等几个问题提出了需求，针对以上的概述，下面从固件实现角度详细分析每个需求的特性。

## 固件驱动层需求分析

EDR读取设备需要建立UDS通讯以及上下位机通讯，其中系统启动后以及通讯时需要有指示灯提示，需要设置以下驱动：

* CAN驱动模块：产品与EDR读取设备500k波特率通讯；
* UART驱动模块：上下位机256000波特率通讯；
* IO模块：信号指示灯提示；
* Timer模块：系统各功能运行时序的管理；

## 固件应用层需求分析

EDR读取设备从应用功能的角度需要实现UDS通讯服务以及上下位机通讯服务，其中需要涉及以下模块：

* UDS功能物理层：CAN数据收发；
* UDS功能数据链路层：发送数据前帧头标识映射封装与接收数据解析；
* UDS功能网络层：接收解析数据后处理(数据拆分、错误处理、超时处理)、发送数据封装、ST时间设置；
* UDS功能应用层：服务区分与UDS服务的解析封装；
* UART功能：数据收发、数据解析、数据封装、数据异常处理(超时处理、错误处理)
* 设备加密功能：EDR读取设备通过UDS服务识别项目唯一性标识，并将结果反馈至上位机显示；

# 系统概念设计

根据“NV1x项目EDR读取设备需求与开发方案”的描述，EDR读取设备建立上下位机的通讯，系统设计的概念框图如下所示：



图 1 EDR读取设备系统框图

如上所述的系统框图，其中EDR读取设备内部包含UART通讯模块、CAN通讯模块、IO模块，根据上位机类型的不同分为USB-TTL模块，外部由OBD母端(OBD端口用于建立CAN通路及12V供电电压)、USB线束、上位机设备等模块组成EDR读取设备系统。

根据“NV1x项目EDR读取设备需求与开发方案”的描述，EDR读取设备整车端的使用流程图如下所示：



图 2 设备使用操作流程图

如上图所示，蓝色模块为EDR设备工装相关的执行动作，绿色模块为EDR上位机相关的操作示例，由此流程图可知，EDR读取设备主要负责EDR相关数据的读取与传递，EDR上位机主要负责EDR相关数据的转译与数据可视化、人机交互等任务。

由图3可知，EDR读取设备功能为条件/事件触发类型，即除自身上电运行外，相关动作的执行均由上位机下发指令后，进行数据读取，数据反馈。

# 软件架构设计

根据第2章节的需求分析与第3章节所描述的系统的组成、设备的操作流程可知，EDR读取设备运行由各个任务触发的逻辑链组成，其中需要固件实时性好，响应及时，根据此特征系统导入类汽车电子专属操作系统OSEK OS，同时配合MCU ISR完成整个固件的软件系统的调度。

软件设计模型和架构如下：

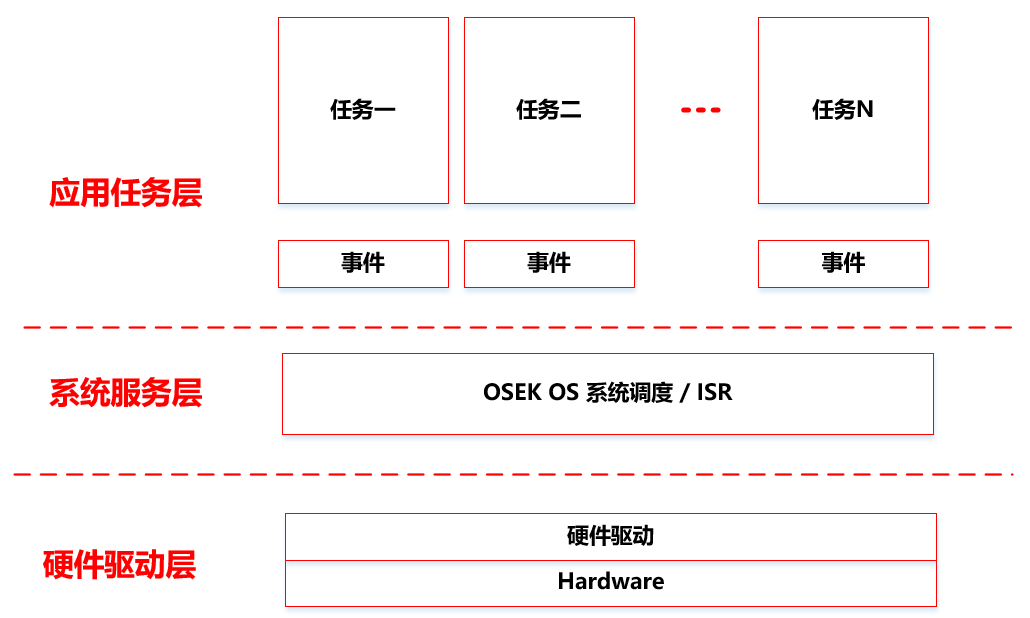


图 3 软件架构模型示意图

其中整个系统被分为三个模块：硬件驱动层、系统服务层与应用任务层，三个模块职能相对独立，协同工作，构建了EDR读取设备的运行机制。

## 硬件驱动层概述

软件系统驱动层代码，主要是封装硬件平台操作信息，为上层系统和应用代码，提供硬件操作统一接口。驱动层代码为上层系统调度和应用代码，屏蔽了底层硬件平台细节信息。从而让上层软件开发，更多专注于软件逻辑和软件功能实现。

驱动层代码和硬件信息高密度耦合。不同的硬件平台，有不同驱动层代码与之匹配。

EDR读取设备驱动层功能模块如下：



图 4 硬件驱动示意图

如上图所示，EDR读取设备基于NXP S32K144芯片进行固件开发设计，其中驱动层被用到5个主要模块，分别为CAN通讯模块、UART通讯模块、FTM定时器模块、 外部输出输入控制信号模块以及INT中断系统模块。

## 系统服务层概述

固件系统主要导入OSEK OS中两大部件：任务调度器和警报器。

EDR读取设备软件系统，将所有的软件功能分别封装成独立的一个个任务（一个任务就是一个没有返回值，完成执行特定功能的函数代码块）。根据每个任务的重要性排序，每一个任务都拥有一个唯一优先级数字；一个优先级下，只有一个任务。OSEK系统调度器，根据任务优先级和任务是否Ready状体，来决定当下执行任务是哪一个。

**OSEK系统调度具有如下特点：**

* OSEK系统调度器，最大支持64个任务。对应优先级为 0 ~ 63。
* 最低优先级为 0， 最高优先级为 63.
* 为支持系统快速响应，任务调度切换事件是1ms. 所有任务都必须在1ms内执行完毕。

**软件系统警报器任务调度：**

EDR读取设备内，包含不少周期性的任务。比如，每隔500ms周期性系统运行指示灯闪烁一次等等。这一类周期性事件任务借由OSEK系统警报器(Alarm)模块调度完成。

所谓软件系统警报器，本质上是一个定时器功能模块。周期性定时事件到达了，就开始去执行与这个事件关联的任务。警报器调度任务方式如下示意：

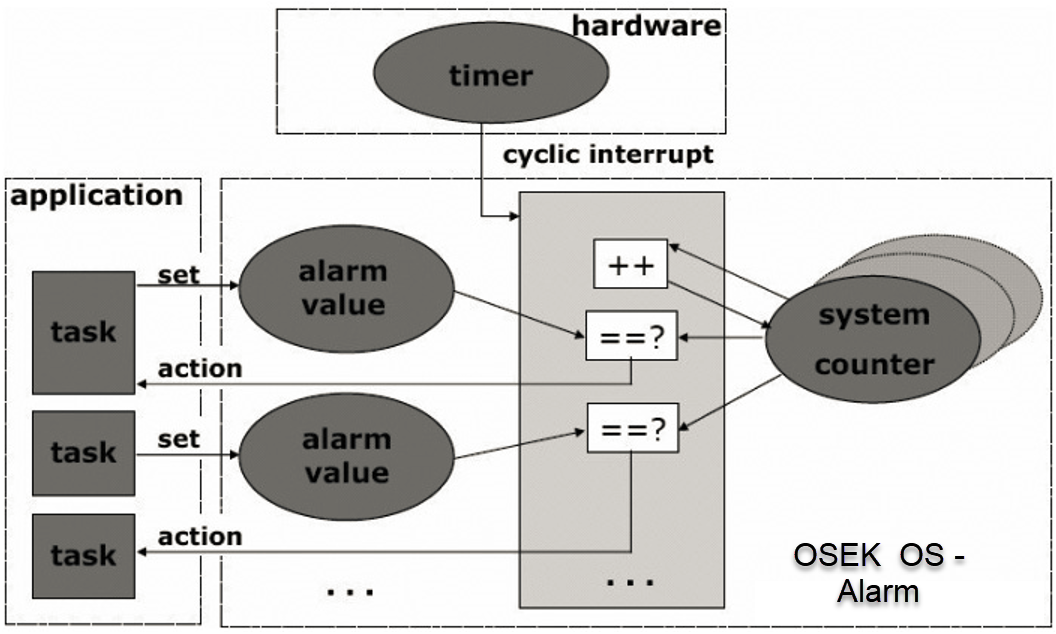


图 5 软件系统警报器调度任务示意图

软件系统的调度由以上两个模块协同运行构成，因此根据其运行机制，可描述系统运行流程图如下所示：



图 5 系统任务运行流程示意图

其中任务调度器为main函数主循环的唯一执行程序。

## 应用任务层概述

EDR读取设备软件系统应用层主要是基于系统需求的软件功能实现任务模块。这些任务模块主要包括诊断通讯模块，上下位机通讯交互模块等等，其中具体的各个任务函数设计在应用层设计模块详细描述。

# 驱动层模块设计

根据第2章节需求描述与第4章节硬件驱动层概述可知，驱动层所述包含五大模块驱动，即External I/O外部PIN脚输入输出信号驱动、CAN通讯模块驱动、UART通讯模块驱动、NVIC中断系统驱动、FTM增强型定时器驱动；各模块驱动的配置如下所示：

## External I/O - 外部输出输入控制信号驱动

MCU的所有信号都是通过IO进行输入输出，其中IO分为两类，一类是GPIO，另一类是功能IO，在此项目中所有的IO设置如下所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **I/O名称** | **I/O定义** | **PIN脚复用** | **初始化状态** | **I/O分类** |
| 系统运行信号灯控制PIN | System\_LED | 硬件原理图异常 | 拉低 | GPIO |
| CAN通讯信号灯控制PIN | CAN\_LED | 硬件原理图异常 | 拉低 | GPIO |
| UART通讯信号灯控制PIN | UART\_LED | 硬件原理图异常 | 拉低 | GPIO |
| CAN数据接收PIN | CAN\_RX | PTE4(5) | 未定义 | 功能IO |
| CAN数据发送PIN | CAN\_TX | PTE5(5) | 未定义 | 功能IO |
| UART数据接收PIN | UART\_RX | PTD6(2) | 未定义 | 功能IO |
| UART数据发送PIN | UART\_TX | PTD7(2) | 未定义 | 功能IO |

上表为基于文件“S32K144\_IO\_Signal\_Description\_Input\_Multiplexing”和原理图描述编写的项目中使用到的7个IO的定义与相应的PIN脚复用设计。

## CAN2.0 A/B通信驱动

CAN通讯模块需要支持500k波特率的通讯速率，根据使用特性CAN一般设计为普通发送和中断接收的功能，因此CAN模块的设计如下所示：

|  |  |
| --- | --- |
| **接口函数** | **配置/功能描述** |
| **void** **FLEXCAN0\_init**()； | 函数为设计函数，设置初始化CAN0模块的统一接口；  函数内部完成CAN功能模块本身的初始化 |
| uint8\_t **FLEXCAN0\_transmit\_msg**(uint8\_t channel,Canframe \* p\_frame) ； | 函数为设计函数，CAN0模块数据发送的统一接口；  其中channel为数据发送设定通道，Canframe 结构体为CAN数据结构体； |
| **void** **FlexCAN0\_MBn\_INT\_enable**(uint8\_t channel)； | 函数为设计函数，CAN0模块的中断初始化 |
| **void CAN0\_ORed\_0\_15\_MB\_IRQHandler (void)**； | 函数为CAN0数据接收中断系统函数 |

以上函数为CAN0模块的驱动相关的主要函数，其中相关硬件配置与逻辑实现参照NV1x项目CAN0模块的配置描述。

## UART通信驱动

UART通讯模块需要支持256000波特率的通讯速率，根据使用特性UART一般设计为中断发送和中断接收的功能，因此UART模块的设计如下所示：

|  |  |
| --- | --- |
| **接口函数** | **配置/功能描述** |
| **void** **LPUART2\_init**(**void**)； | 函数为设计函数，设置初始化UART2模块的统一接口；  函数内部完成UART2功能模块本身的初始化 |
| **void** **LPUART2\_RxTx\_INT\_enable**(**void**)； | 函数为设计函数，UART2模块的中断初始化 |
| **void** **LPUART2\_RxTx\_IRQHandler**(**void**)； | 函数为UART2数据接收与接收中断系统函数 |

以上函数为UART模块的驱动相关的主要函数，其中相关硬件配置与逻辑实现参照NV1x项目产线滑台工装UART模块的配置描述。

## FTM灵活定时器驱动

FTM模块需要支持250us的中断触发，根据FTM使用特性，FTM模块的设计如下所示：

|  |  |
| --- | --- |
| **接口函数** | **配置/功能描述** |
| **void** **FTM0\_init**(**void**)； | 函数为设计函数，设置初始化FTM0模块的统一接口；  函数内部完成FTM功能模块本身的初始化 |
| **void** **FTM0\_CH0\_OC\_init**(**void**)； | 函数为设计函数，功能实现FTM0通道0输出比较初始化，其中需要注意打开中断使能 |
| **void** **FTM0\_INT\_enable**(uint8\_t channel)； | 函数为设计函数，FTM0模块的中断初始化 |
| **void** **FTM0\_Ch0\_Ch1\_IRQHandler**(**void**)； | 函数为FTM0通道0时间触发中断系统函数 |

以上函数为FTM0模块的驱动相关的主要函数，其中相关硬件配置与逻辑实现参照NV1x项目FTM0模块的配置描述。

# 应用层模块设计

根据第2章节需求描述与第4章节软件架构设计可知，EDR读取设备应用层模块是按照任务形式设计，每个任务对应的是一个没有返回值的函数，实现一定的功能逻辑，因此应用层按照UDSonCAN通讯模块、UART通讯模块、设备加密模块等几大模块，每个模块由一个或多个任务协同工作以实现功能逻辑，详细的功能模块的设计如下描述：

## UDSonCAN模块应用设计

EDR读取设备获取ACU/EDR产品相关信息均通过UDS服务访问，根据功能需求，EDR读取设备需要实现的功能及对应的UDS服务如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UDS服务** | | | | |
| **服务标识符** | **诊断服务名称** | **DID/子功能 (Hex)** | **功能描述** | **服务请求 (Hex)** |
| $22 | 读取数据 | F18C | ECU序列号 | 03 22 F1 8C 00 00 00 00  30 00 0A 00 00 00 00 00 |
| F193 | ECU硬件版本号 | 03 22 F1 93 00 00 00 00  30 00 0A 00 00 00 00 00 |
| F195 | ECU软件版本号 | 03 22 F1 95 00 00 00 00  30 00 0A 00 00 00 00 00 |
| F18B | ECU生产日期 | 03 22 F1 8B 00 00 00 00 |
| F101 | 功能配置字 | 03 22 F1 01 00 00 00 00 |
| FA13 | FA13事件数据 | 03 22 FA 13 00 00 00 00  30 00 0A 00 00 00 00 00 |
| FA14 | FA14事件数据 | 03 22 FA 14 00 00 00 00  30 00 0A 00 00 00 00 00 |
| FA15 | FA15事件数据 | 03 22 FA 15 00 00 00 00  30 00 0A 00 00 00 00 00 |
| 0216 | 0216事件数据 | 03 22 02 16 00 00 00 00  30 00 0A 00 00 00 00 00 |
| 0217 | 0217事件数据 | 03 22 02 17 00 00 00 00  30 00 0A 00 00 00 00 00 |
| 0218 | 0218事件数据 | 03 22 02 18 00 00 00 00  30 00 0A 00 00 00 00 00 |
| 0219 | 0219事件数据 | 03 22 02 19 00 00 00 00  30 00 0A 00 00 00 00 00 |
| $19 | 读取DTC信息 | 02 | 故障信息 | 03 19 02 FF 00 00 00 00  30 00 0A 00 00 00 00 00 |
| $14 | 清除诊断信息 | NA | 清除故障码 | 04 14 FF FF FF 00 00 00 |
| $10 | 诊断会话控制 | 03 | 扩展诊断会话 | 02 10 03 00 00 00 00 00 |
| $27 | 安全访问 | 01 | 请求级别1种子 | 02 27 01 00 00 00 00 00 |
| 02 | 发送级别1密钥 | 06 27 02 XX XX XX XX 00 |
| $2E | 写入数据 | 0233 | 激活内部服务 | 05 2E 02 33 01 55 00 00 |

以上列表为EDR读取设备功能应用所涉及的相关UDS服务内容18项，所涉及功能共计14项(主要涉及$22、$19、$14三项诊断服务)，额外的服务($10、$27、$2E)是为读取$220216 ~ $220219等4项服务数据而激活内部服务使用的；

列表中的ECU序列号、ECU硬件版本号、ECU软件版本号、ECU生产日期、功能配置字、FA13事件数据、FA14事件数据、FA15事件数据、故障信息、清除故障码等10项功能服务可直接通过服务请求完成诊断会话；

列表中的0216事件数据、0217事件数据、0218事件数据、0219事件数据等4项功能服务因为是内部信息，不直接对外开放，因此需要通过扩展诊断会话($1003)、安全访问请求种子($2701)、安全访问发送密钥($2702)、写入数据激活内部服务($2E0233)等四个步骤后，才可以进入相应服务读取ACU碰撞事件数据。

### UDS服务—物理层与数据链路层

**物理层：**

物理层为设备之间的[数据通信](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E9%80%9A%E4%BF%A1/897073" \t "https://baike.baidu.com/item/_blank)提供传输媒体及互连设备，为[数据传输](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E4%BC%A0%E8%BE%93/2987565" \t "https://baike.baidu.com/item/_blank)提供可靠的环境，相当于某种“介质”。这种“介质”的物理介质附件层（PMA）和物理介质依赖层（PMD）是由CAN收发器和双绞线以及终端电阻组成，是整个网络通讯的基础。

**物理层函数列表**

|  |  |
| --- | --- |
| **函数** | **功能** |
| uint8\_t FLEXCAN0\_receive\_msg(uint8\_t channel) | 接收CAN数据帧 |
| uint8\_t FLEXCAN0\_transmit\_msg(uint8\_t channel,Canframe \* p\_frame) | 发送CAN数据帧 |

**数据链路层：**

数据链路层是控制网络层与物理层之间的通信。主要功能是如何在不可靠的物理线路上进行数据的可靠传递。为了保证传输，从网络层接收到的数据将分割成待定的可被物理层传输的帧。如果在传输数据时，接收点检测到所传数据中有差错，就要通知发送方重发这一帧。

**数据链路层函数列表**

|  |  |
| --- | --- |
| **函数** | **功能** |
| UChar Can\_Frame\_Map\_NtD(Buffer \*p\_buffer,Canframe\* frame) | 发送数据前的帧头标识映射封装 |
| void Can\_Receive\_Task(void) | 对接收的数据进行解析 |

**函数功能解析**

* **CAN\_Receive\_Task()函数**

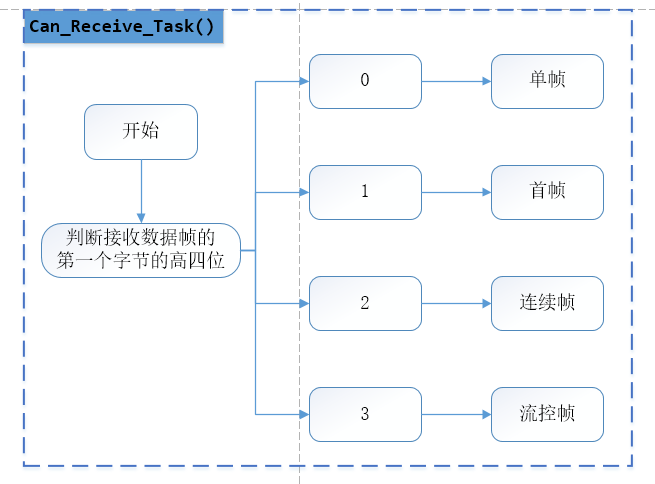
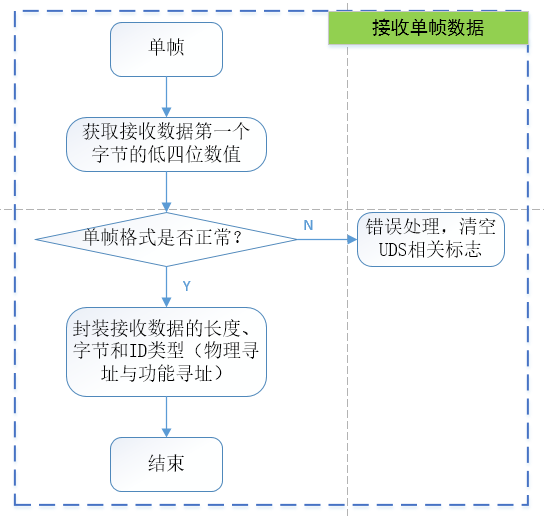
 

图 6 接收函数总流程图 图7 接收单帧数据流程图

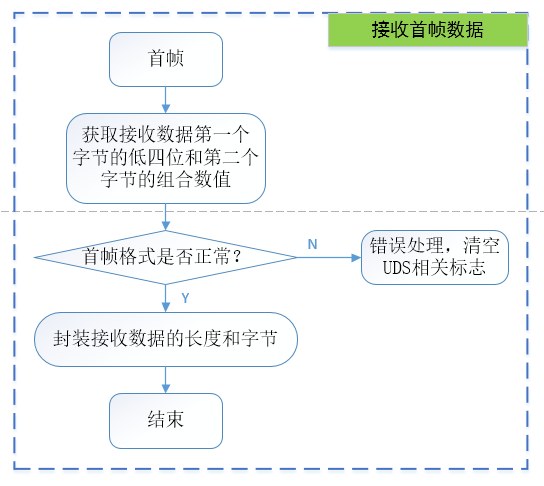
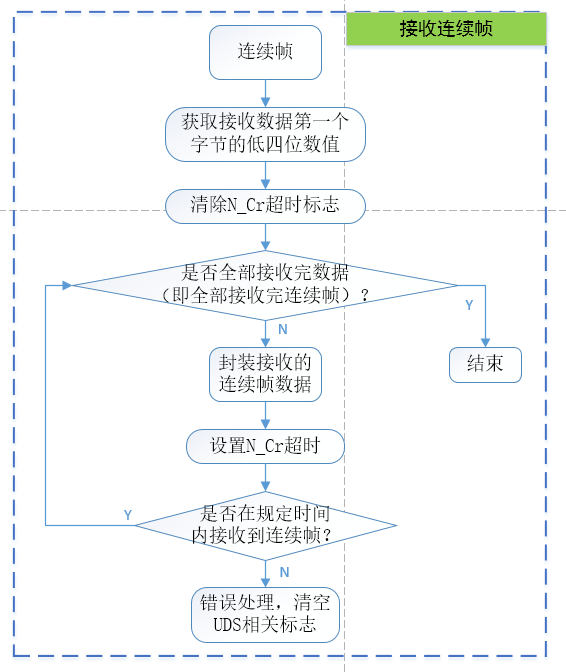
 

图8 接收首帧数据流程图 图9接收连续帧数据流程图

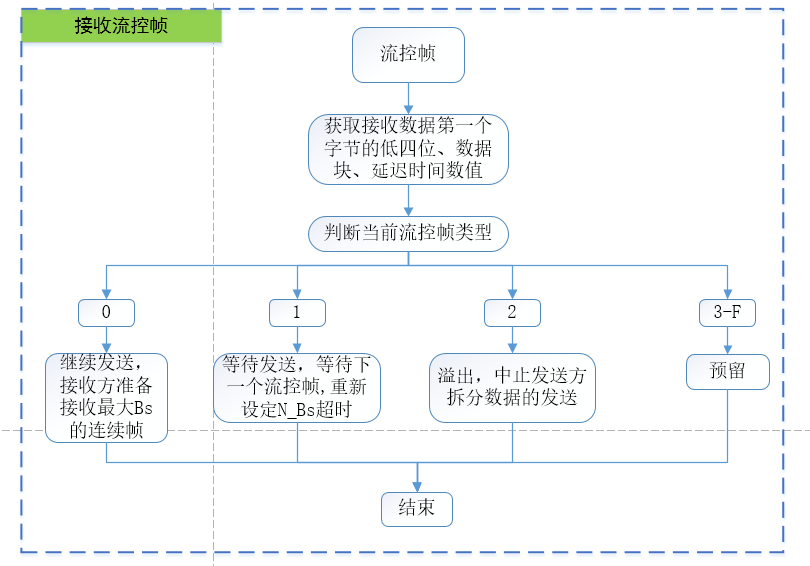


图10接收流控帧数据流程图

**分析**

1. CAN\_Receive\_Task()函数作用是对物理层接收到的CAN数据进行拆分，封装。
2. 接收到的CAN数据分为单帧、首帧、连续帧和流控帧四种帧类型。
3. 单帧数据的第一个字节后四位表示该数据帧的长度。长度不合要求的进行错误处理，结束该轮通信；长度符合规定的数据，封装数据长度、字节数据和标识符信息。
4. 首帧数据的第一个字节的低四位和第二个字节的组合表示该数据帧的长度。长度不合要求的进行错误处理，结束该轮通信；长度符合规定的数据，封装数据长度和字节数据信息。
5. 连续帧数据的第一个字节的后四位表示连续帧数据的连续性。不连续的数据进行错误处理，结束该轮通信；格式正常的连续帧根据长度全部接收才可结束接收，未接收全数据，需在接收每一帧连续帧后设定N\_Cr（接收连续帧时间超时）计时，若在规定超时内未及时接收连续帧，该轮通信结束。
6. 流控帧数据的第一个字节的后四位表示当前流控帧的类型。“0”：表示继续发送，接收方将接收全部大小的连续帧。“1”：表示等待，重新设定N\_Bs计时器，等待接收下一个流控帧。“2”：表示溢出，接收该轮通信。

### UDS服务—网络层

UDS网络层其存在的目的是为了解决ISO 11898协议中定义的经典CAN数据链路层与ISO 14229协议中定义的应用层，彼此之间数据长度不统一的问题。经典CAN数据链路层最大能够支持8个字节，但ISO 14229并不仅仅是为了CAN总线设计的，最大容量达到4095个字节。比如VIN码（车辆代号）是17个字节，CAN总线必然需要传递3帧才能传完VIN码，那么如何科学、快捷、安全地将多个字节通过经典CAN来进行传输，就成了一个需要解决的问题，网络层（ISO 15765-2 协议）主要就是解决这个CAN网络数据传输问题。

网络层是为传输层提供服务的，把数据转换成能适应CAN总线规范的单一数据帧，从而进行传输。如果将要传输的报文长度超过了CAN数据帧的长度，则需要将报文信息进行拆分后传输，每次至多可以传输4095个字节长度的报文。因此，网络层CAN数据传输有两种数据形式：单帧传输，多帧传输。

**网络层函数列表**

|  |  |
| --- | --- |
| **函数** | **功能** |
| void Can\_Request\_Function\_Task(void) | 对接收并解析过的数据进行拆分 |
| void FlowControlData(Buffer \*buf,Flowtype type) | 对发送的流控帧数据进行封装 |
| void Can\_Response\_Function\_Task(void) | 对回应的数据进行拆分处理 |
| void SplitData(Buffer \*buf,UChar len,Flowtype typ) | 对单帧传输、多帧传输的数据进行拆分 |
| void Error\_Handle\_Function(ErrorFlag \* ErrorPoint) | 对解析出错的数据处理 |
| void Can\_timeout\_task(void) | 对未及时接收到帧数据进行超时处理 |
| UChar AcquireSTminTime(UChar Value) | ST的时间设置 |

**函数功能解析**

* **CAN\_Request\_Function\_Task()函数**

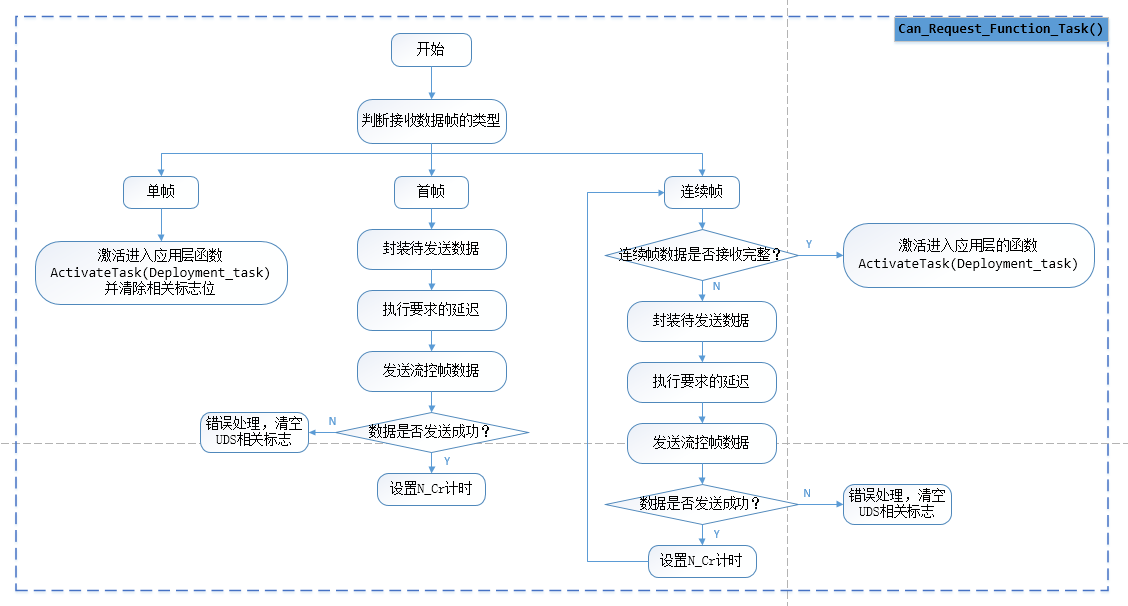


图11对解析过的接收数据进行拆分

**分析**

1. 拆分接收数据的类型包括单帧、首帧、连续帧。
2. 接收单帧数据，进入应用层判断该单帧数据的功能和封装响应数据。
3. 接收首帧数据，进行规范中的延迟后，响应流控帧给tester，响应出错进行错误处理，结束该轮通信；响应正常，设定N\_Cr计时，等待tester在规定时间内发送连续帧。
4. 接收连续帧数据，首先判断连续帧数据是否完整，接收完整进入应用层判断该单帧数据的功能和封装响应数据；未接收完整则响应流控帧，并设定N\_Cr计时，等待连续帧的发送，直至接收完整连续帧，超时则结束该轮通信。

* **CAN\_Response\_Function\_Task()函数**

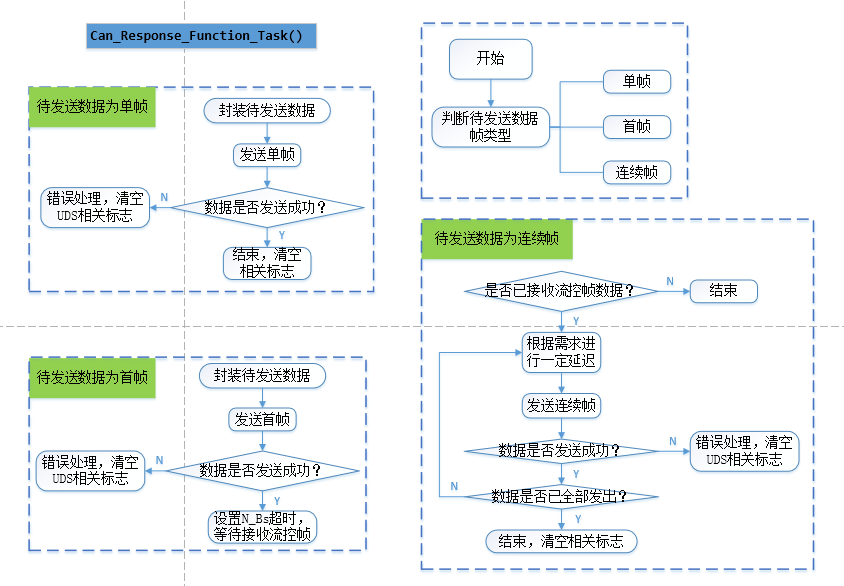


图12对回应的数据进行拆分处理并执行发送函数

**分析**

1. CAN\_Response\_Function\_Task()函数用于对响应的数据进行拆分处理并执行发送函数。
2. 处理的数据类型分为单帧、首帧和连续帧。
3. 待发送数据为单帧时，封装数据，并判断数据是否发送成功。
4. 待发送数据为首帧时，封装数据，并判断数据是否发送成功。对发送成功的数据设置N\_Bs计时，等待接收流控帧；对发送失败进行错误处理并清空相关UDS标志。
5. 待发送数据为连续帧时，判断是否已接收过流控帧数据，未接收过则不发送连续帧；若接收过流控帧数据，则根据流控帧内的时间间隔发送连续帧，直至连续帧全部发完。

* **延迟时间参数**

|  |  |
| --- | --- |
| **Tester发送流控帧中时间（HEX）** | **Server** |
| 00 | 1 ms |
| 1 - 7F | 1 - 127 ms |
| F1 - F9 | 100 - 900 us |
| 其余（最大FF） | 127 ms |

### UDS服务—应用层

应用层是OSI参考模型中的最高层,是直接为应用进程提供服务的。其作用是在实现多个系统应用进程相互通信的同时,完成一系列业务处理所需的服务。

**应用层函数列表**

|  |  |
| --- | --- |
| **函数** | **功能** |
| void Can\_Deployment\_Task(void) | 对访问的服务进行区分 |
| void task\_UDS\_DataProcess(ErrorFlag errType) | UDS服务的解析与封装 |

**应用层功能模块**

* **请求与回应格式分类**

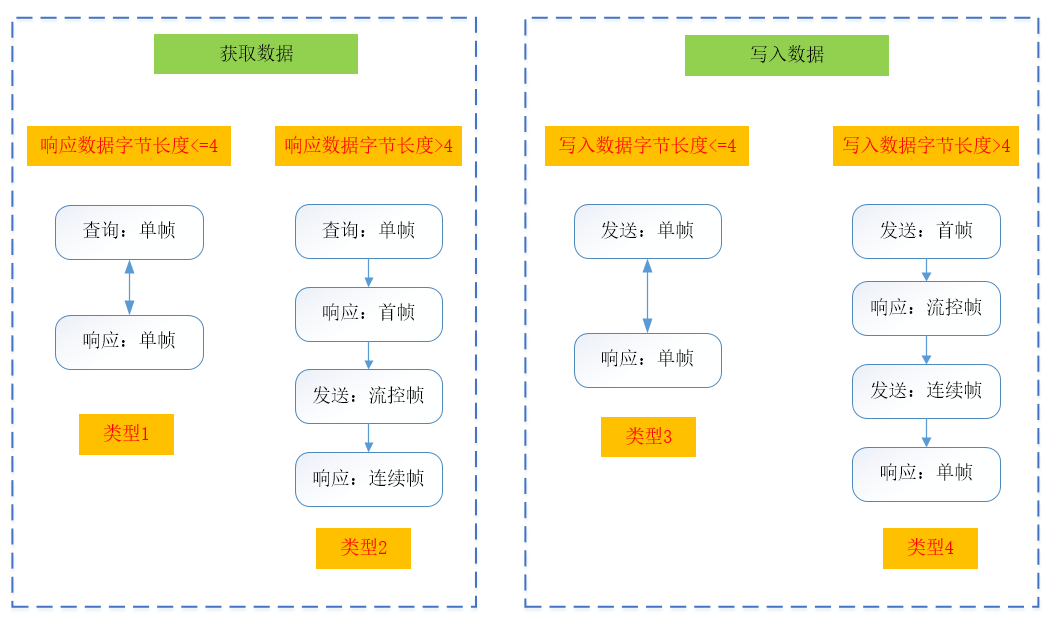


图13数据请求与响应格式图解

UDS服务主要的功能分为获取数据与写入数据两个功能，在满足正响应的格式下，其中每个功能都包括两个类型的格式。

* **获取数据**

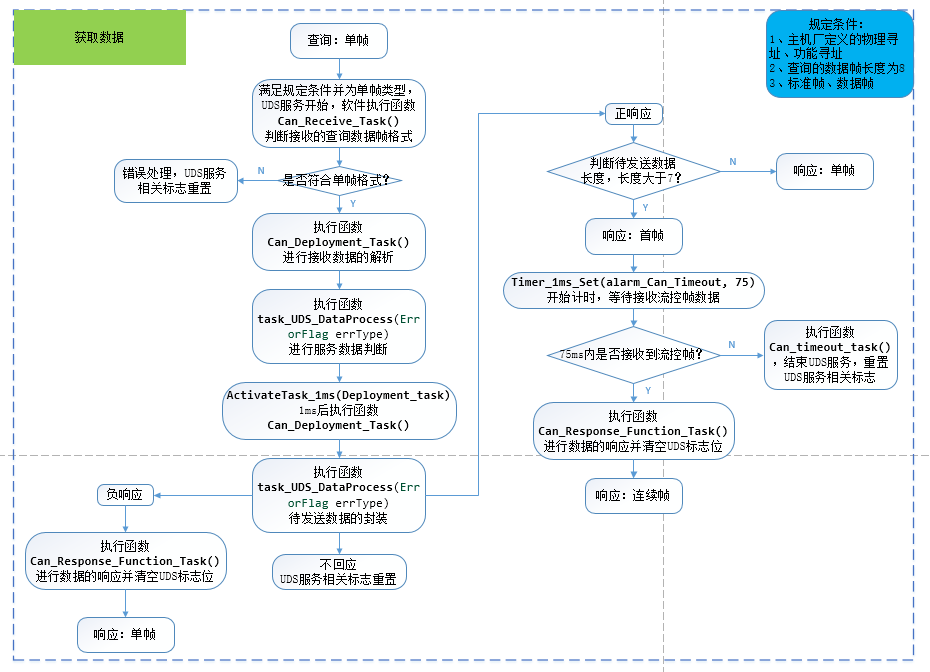


图14获取数据流程图解

**分析**

1. 获取数据模块，满足主机厂定义的物理寻址、功能寻址、查询的数据帧长度为8bytes以及CAN数据符合标准帧与数据帧。
2. Tester方发送单帧的请求服务，Server判断数据格式是否正确，格式错误进行错误处理结束该轮服务。
3. 格式正确的数据进行数据拆分，判断获取数据的服务参数，对回应的格式进行确认。
4. 响应分为正响应、负响应和不回应三种格式。不回应的格式结束该轮服务，清空相关参数标志。负响应的格式需要给出对应的否定响应码，并响应的数据为单帧格式。
5. 正响应的格式，需要对待发送的数据长度进行分析。数据长度大于7，则以首帧格式响应；反之，则以单帧格式回应。
6. Server响应首帧后，设定N\_Bs计时，Tester需要在规定的时间内，发出流控帧请求。对于超时未发送的UDS服务，结束该轮UDS服务，重置相关服务标志，等待下一轮服务。规定时间内发出流控帧请求，Server响应连续帧，直至连续帧全部发出。

* **写入数据**

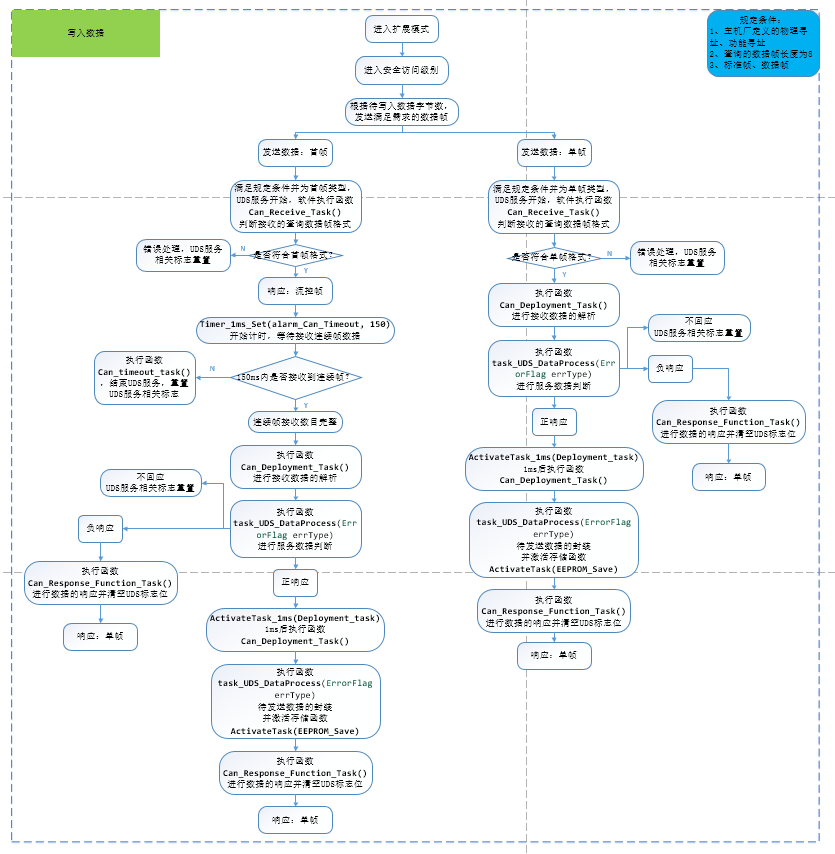


图14写入数据流程图解

**分析**

1. 写入数据模块，满足主机厂定义的物理寻址、功能寻址、查询的数据帧长度为8bytes以及CAN数据符合标准帧与数据帧。
2. 写入数据前必须进入安全访问等级。根据写入数据的长度，写服务分为单帧格式和首帧格式。
3. 响应的格式分为：正响应、负响应和不响应。执行写操作只能在正响应的前提下。正响应成功后执行向存储器记录待存储的数据函数。
4. 对于首帧格式的写服务请求，响应流控帧后，设置N\_Cr计时，等待接收连续帧数据。在规定时间内未接收到连续帧，执行超时处理，清除UDS相关服务标志。
5. 在规定时间内接收到连续帧数据后，需判断是否接收完整的连续帧。对于未接收完整的连续帧，重新设定N\_Cr计时，继续接收。接收完整后，关闭N\_Cr计时并清空相关UDS服务标志。

* **数据标识符的自由切换**

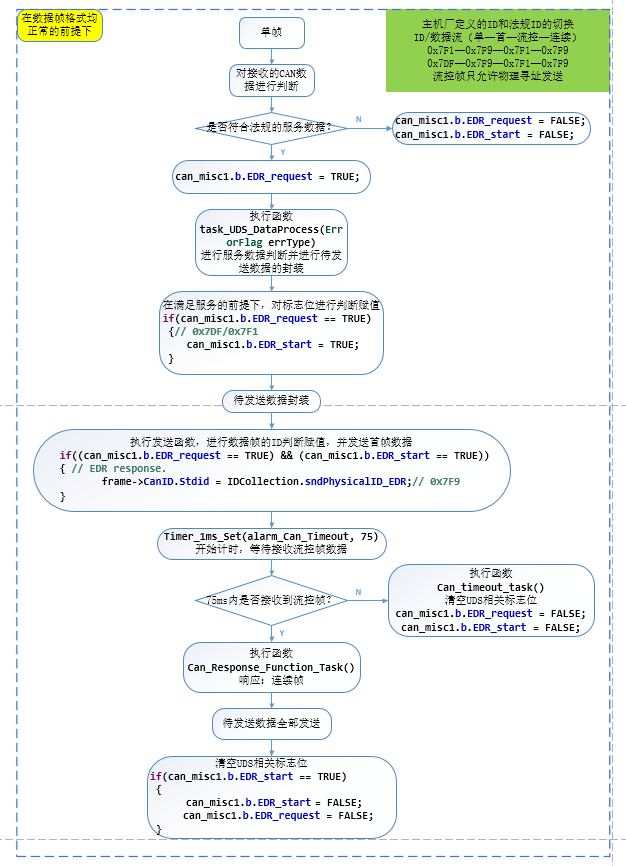
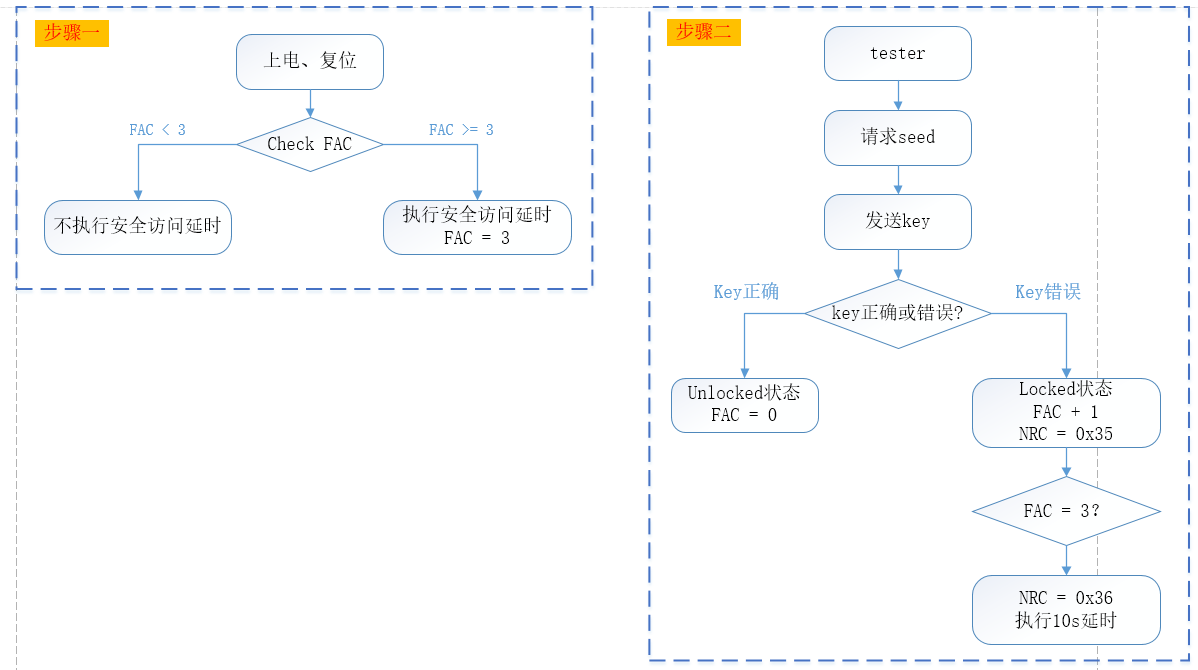


图15数据标识符自由切换流程图解

**分析**

1. 通过设定两个标志，实现UDS服务中主机厂定义与法规定义标识符的切换功能。
2. 对于接收的数据满足法规规定，置位其中一个标志。
3. 判断拆分后的数据是否满足规定的服务格式，对于满足规定的格式置位另一个标志。
4. 在数据标识符的封装时，进行两个标志位是否均置位的判断，均置位则标识符为法规规定的。

* **安全访问机制**



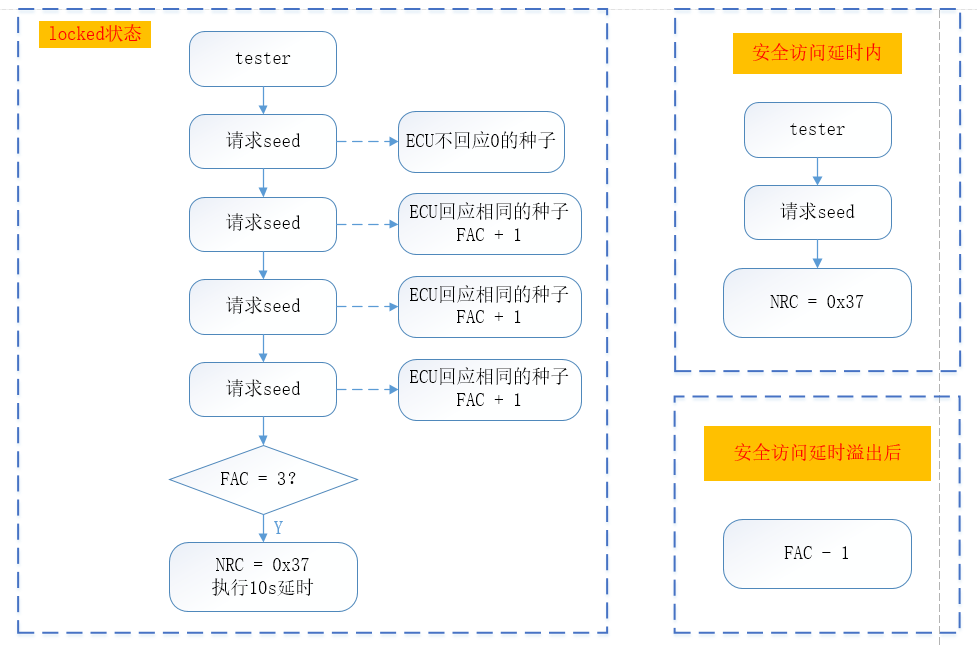
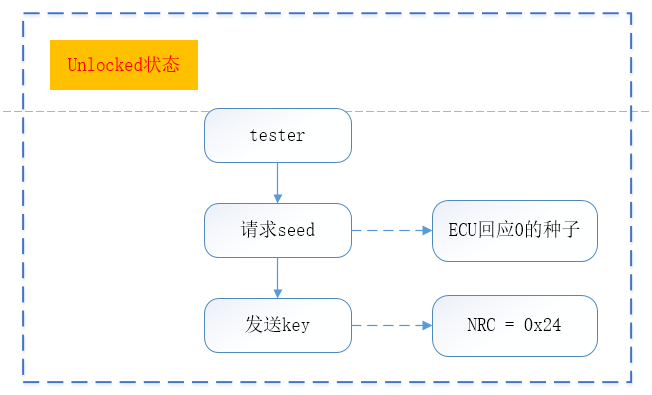


图16安全访问机制流程图解

**分析**

1、ECU 应记录并保存 FAC（Failure Attempt Counter），推荐将 FAC 保存到 EEPROM，保存时机由各个 ECU 决定，但必须保证在 ECU 重启（复位或下上电）后不会丢失 FAC。

2、上电、复位后，ECU 首先检查各个安全等级的 FAC，如果 FAC 小于 3，ECU 无需执行安全访问延时，反之，应该执行安全访问延时（延时时间为 10s），并将 FAC 赋值为 3。

3、安全访问应该遵循“Seed-Key” 序列，即 tester 发送一次“request Seed”，接着发送一次Key。如果 tester 发送了正确的 key，相应的安全等级进入 Unlocked 状态，同时 FAC 赋值为 0。如果 tester 发送了错误的 key，将导致 FAC 加 1，同时 ECU 回复 NRC 0x35，如果由于错误的 key 导致 FAC 等于 3， ECU 回复 NRC 0x36 的否定响应之后再执行 10s 延时。

4、某安全等级处于 Locked 状态，如果 tester 发送“request seed”，ECU 不应该发送值为 0 的种子。如果 tester 连续发送“request seed”，ECU 应该返回相同的 seed，但 FAC 应该加 1，如果由于连续发送“request seed”导致 FAC 等于 3（即第四次连续请求种子），ECU 回复 NRC 0x37 的否定响应之后再执行 10s 延时。

5、如果 ECU 在安全访问延时未溢出之前收到“request Seed”请求，应回复 NRC 0x37 的否定响应。安全访问延时溢出后 FAC 减 1，即允许一次安全访问。

6、某安全等级处于 Unlocked 状态，如果 ECU 收到“request Seed”请求，ECU 应回复值为 0 的种子，此时，如果 tester 继续发送 Key，ECU 应回复 NRC 0x24 的否定响应。

7、当 ECU 被重新上电、复位或执行会话跳转，“Seed-Key” 序列将被重置。

8、种子为除0x00000000和0xFFFFFFFF数值以外的随机数。

## UART通讯模块应用设计

UART通讯模块核心功能是通过建立通讯，将上位机的需求传递至下位机，并将产品信息反馈至上位机；以此为UART的重要功能模块就要求明确上位机的功能需求，并且合理高效反馈产品对外表达的信息，则UART相关的通讯模块设计详细描述如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UART通讯交互内容** | | | | |
| 项目匹配 |  |  |  |  |
| ECU序列号 | ECU硬件版本号 | ECU软件版本号 | ECU生产日期 | 功能配置字 |
| FA13事件数据 | FA14事件数据 | FA15事件数据 |  |  |
| 0218事件数据 | 0219事件数据 | 0216事件数据 | 0217事件数据 |  |
| 故障信息 | 清除故障码 |  |  |  |

如上表可知UART通讯交互内容共4类15项，其中3类14项均与UDS诊断通讯内容一致，针对这一部分内容就是将UDS数据转译成UART数据进行上下位机交互，还有1项是设备加密所涉及的项目匹配信息，此信息为最高优先级交互信息，如果项目不匹配，则上下位机所有后续交互信息无效。

根据串口通讯公司统一要求“串口总线（485）通信协议规范v1.5.DOCX”所描述，当前通讯机制设计涉及如下：

### 串口帧格式规范



**帧头：**大小2字节，建议数值采用0xAAAA；

**长度码：**大小1字节、可表示数值范围0~255。长度码是一帧除去帧头外的长度；

**从设备地址：**大小1字节、可表示数值范围0~255；

**数据：**大小可变、可表示数值范围0~255；

**差错校验：**采用**累加和**校验校验；

注意：数据包**帧头**域的第一个字节算起，**数据**域为结尾。见下图



### 串口通讯协议内容

串口通讯协议内容为6.2.1章节所描述的数据段内容，主要是参照产品产线的协议制定思路，根据UDS诊断通讯协议设定相应的通讯内容，其中主要满足UART通讯交互内容，详细的协议制定如下所示：

**协议格式**

|  |  |  |  |
| --- | --- | --- | --- |
| **UART通信协议格式定义** | | | |
| **字节1** | **字节2** | **字节3** | **字节4 - 字节N** |
| 一级识别码  ***初始值：0x00*** | 二级识别码  ***初始值：0x00*** | 三级识别码  ***初始值：0x00*** | 状态码  ***初始值：0x00*** |

**协议内容(上位机发送)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UART通信协议内容定义*（上位机发送）*** | | | | |
| **一级识别码**  **（1 Byte）** | **二级识别码**  **（1 Byte）** | **三级识别码**  **（1 Byte）** | **状态码** | **通信含义** |
| 0xAA | 0xAA | 0xAA | N/A | 项目匹配信息 |
| 0x22 | 0xF1 | 0x8C | N/A | ECU序列号 |
| 0x22 | 0xF1 | 0x93 | N/A | ECU硬件版本号 |
| 0x22 | 0xF1 | 0x95 | N/A | ECU软件版本号 |
| 0x22 | 0xF1 | 0x8B | N/A | ECU生产日期 |
| 0x22 | 0xF1 | 0x01 | N/A | 功能配置字 |
| 0x22 | 0xFA | 0x13 | N/A | FA13事件数据 |
| 0x22 | 0xFA | 0x14 | N/A | FA14事件数据 |
| 0x22 | 0xFA | 0x15 | N/A | FA15事件数据 |
| 0x22 | 0x02 | 0x16 | N/A | 0216事件数据 |
| 0x22 | 0x02 | 0x17 | N/A | 0217事件数据 |
| 0x22 | 0x02 | 0x18 | N/A | 0218事件数据 |
| 0x22 | 0x02 | 0x19 | N/A | 0219事件数据 |
| 0x19 | 0x02 | 0xFF | N/A | 故障信息 |
| 0x14 | 0xFF | 0xFF | N/A | 清除故障码 |

**协议内容(下位机反馈)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UART通信协议内容定义*（下位机发送）*** | | | | |
| **一级识别码**  **（1 Byte）** | **二级识别码**  **（1 Byte）** | **三级识别码**  **（1 Byte）** | **状态码** | **通信含义** |
| 0xAA | 0xAA | 0xAA | **字节1：**匹配标志位  (0x55: 成功 0xAA: 失败  0x00: 数据获取中) | **项目匹配信息** |
| 0x22 | 0xF1 | 0x8C | **字节1：**0x00(数据获取中) | **ECU序列号** |
| 0x62 | 0xF1 | 0x8C | **字节1-64：**ASCII码  (字节数小于 64时，应用 0x20 填充，填充位在左边) |
| 0x22 | 0xF1 | 0x93 | **字节1：**0x00(数据获取中) | **ECU硬件版本号** |
| 0x62 | 0xF1 | 0x93 | **字节1-64：**ASCII码  (字节数小于 64时，应用 0x20 填充，填充位在左边) |
| 0x22 | 0xF1 | 0x95 | **字节1：**0x00(数据获取中) | **ECU软件版本号** |
| 0x62 | 0xF1 | 0x95 | **字节1-64：**ASCII码  (字节数小于 64时，应用 0x20 填充，填充位在左边) |
| 0x22 | 0xF1 | 0x8B | **字节1：**0x00(数据获取中) | **ECU生产日期** |
| 0x62 | 0xF1 | 0x8B | **字节1-4：**BCD码  (年：2字节/月：1字节/日：1字节) |
| 0x22 | 0xF1 | 0x01 | **字节1：**0x00(数据获取中) | **功能配置字** |
| 0x62 | 0xF1 | 0x01 | **字节1：**功能配置字  (0x01: 正常模式 0x02: 工厂模式) |
| 0x22 | 0xFA | 0x13 | **字节1：**0x00(数据获取中) | **FA13事件数据** |
| 0x62 | 0xFA | 0x13 | **字节1-772：**依据法规CAN格式解析  (数据分为4次发送) |
| 0x22 | 0xFA | 0x14 | **字节1：**0x00(数据获取中) | **FA14事件数据** |
| 0x62 | 0xFA | 0x14 | **字节1-772：**依据法规CAN格式解析  (数据分为4次发送) |
| 0x22 | 0xFA | 0x15 | **字节1：**0x00(数据获取中) | **FA15事件数据** |
| 0x62 | 0xFA | 0x15 | **字节1-772：**依据法规CAN格式解析  (数据分为4次发送) |
| 0x22 | 0x02 | 0x16 | **字节1：**0x00(数据获取中) | **0216事件数据** |
| 0x62 | 0x02 | 0x16 | **字节1-1424：**依据ACU格式解析  (数据分为6次发送) |
| 0x22 | 0x02 | 0x17 | **字节1：**0x00(数据获取中) | **0217事件数据** |
| 0x62 | 0x02 | 0x17 | **字节1-1424：**依据ACU格式解析  (数据分为6次发送) |
| 0x22 | 0x02 | 0x18 | **字节1：**0x00(数据获取中) | **0218事件数据** |
| 0x62 | 0x02 | 0x18 | **字节1-976：**依据ACU格式解析  (数据分为4次发送) |
| 0x22 | 0x02 | 0x19 | **字节1：**0x00(数据获取中) | **0219事件数据** |
| 0x62 | 0x02 | 0x19 | **字节1-976：**依据ACU格式解析  (数据分为4次发送) |
| 0x19 | 0x02 | 0xFF | **字节1：**0x00(数据获取中) | **故障信息** |
| 0x59 | 0x02 | 0xFF | **字节1-64：**依据UDS故障码格式  (故障码最大16个，未满16个时，剩下的故障码均填充0xFFFFFFFF) |
| 0x14 | 0xFF | 0xFF | **字节1：**0x00(数据获取中) | **清除故障码** |
| 0x54 | 0xFF | 0xFF | **字节1：**故障码清除标志位  (0x55: 成功 0xAA: 失败) |

### 串口通讯功能函数设计

串口通讯相比UDSonCAN诊断通讯，实现逻辑较为简单，其接收一次上位机的通讯，并进行一轮通讯回复，其中需要注意的是，由于系统调度的时序要求和满足串口通讯的长数据发送，串口通讯接收发送统一采用中断模式，详细的设计细节如下：

|  |  |
| --- | --- |
| **接口函数** | **配置/功能描述** |
| **void** **task\_SCI\_TX**(**void**)； | 串口数据发送逻辑函数，用于激活串口数据发送 |
| **void** **LPUART2\_RxTx\_IRQHandler**(**void**)； | 系统中断函数，函数内实现串口数据的收发逻辑 |

**函数解析：void** **task\_SCI\_TX**(**void**)

****

图17串口发送函数逻辑框图

**函数解析：void** **LPUART2\_RxTx\_IRQHandler**(**void**)

****

图18串口中断函数逻辑框图

串口通讯的数据收发功能主要由以上两个函数的逻辑实现，串口通讯由于主从关系，必然是由上位机下发数据开始，下位机反馈数据为逻辑闭环结束，因此数据发送的激活均在接收完成后立即反馈的，反馈数据的封装在UDSonCAN通讯完成后进行的，反馈数据基本为UDS数据转译。

## 设备加密模块设计

设备加密功能旨在设备本身的适配性和设备使用的专属性，从功能的可靠性和功能实现的便捷性出发，加密功能可识别项目唯一性标识：软件版本号(软件版本号是针对每个项目特殊性的编号)，当上位机询问项目匹配信息时，下位机则通过UDS服务$22F195进行诊断会话，收到数据后和确定的编号核对校验，当数据完全匹配时，反馈项目匹配成功，否则反馈项目匹配失败；



图19设备加密逻辑框图

# 设计总结

本报告设计依据需求，从固件方面详细描述EDR读取设备实现的细节，其中分别从软件架构(OSEK类操作系统)、驱动层(IO、CAN、UART、FTM)、应用层(UDSonCAN诊断模块、UART通讯模块、加密功能)的实现进行设计，本次设计目标实现总体方案中描述的“1阶段实现串口通讯+PC端上位机”，固件根据后续代码编译与测试过程对本文档进行迭代更新。