文章编号：2095-2163(2021)01-0102-04 中图分类号：TP26 文献标志码：A

基于 UDS 协议的汽车控制器刷写软件设计

唐恒飞[1]，王效金[2]

（1 上海工程技术大学 机械与汽车工程学院，上海 201620；2 上海银基信息安全技术股份有限公司，上海 200120）

摘 要： 基于某整车厂提供的控制器和刷写程序（.hex 文件），利用实验室研发的汽车诊断仪，设计了一款上位机刷写软件。该软件是利用 C#语言中的 WinForm 界面和基于 CAN 总线的 UDS 协议进行编写，并且遵循软件设计 V 型开发流程，主要内容是刷写方案的设计，以及研究上位机软件与控制器之间信息的交互，最后该软件经过上下位机联调测试，可以很好地体现其正确性以及可靠性。

关键词： 刷写；UDS 协议；C#；ECU；V 型

Design of flash writing software for automobile controller based on UDS protocol

TANG Hengfei[1], WANG Xiaojin[2]

(1 School of Mechanical and Automotive Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; 2 Shanghai Yinji Information Security Technology Co. Ltd., Shanghai 200120, China)

【Abstract】 Based on the controller and flash writing program (.hex file) provided by a vehicle factory, a flash writing software for upper computer is designed by using the automobile diagnosis instrument developed in the laboratory. The software is written by WinForm interface in C# language and UDS protocol based on CAN bus, and follows the V-shaped development process of software design. The main content is the design of brush writing scheme and the information interaction between the upper computer software and the controller. Finally, the software has been tested by the upper and lower computers, which can reflect its correctness and reliability.

【Key words】 flash and write; UDS protocol; C#; ECU; V type

0 引 言

随着互联网技术的迅猛发展，带动整个汽车行业加速走向了智能化。由此即提升了汽车控制单元的控制策略和功能复杂度，进而使得控制器内部软件更新升级的要求也越来越高。因此国内外有众多的学者和企业都在致力于研究汽车控制器刷写。文献[1]基于 UDS 协议和 MPC5634 单片机，设计了一款在线升级软件；文献[2]利用 LabView 编程语言，基于 UDS 协议设计了 ECU 刷写上位机软件；文献[3]提出了一种基于 UDS 协议的汽车 ECU 升级方案，详细讨论了汽车 ECU 的 BootLoader 刷写软件的实现原理。本文则是利用 C#语言，基于 UDS 协议和 CAN 通讯机制设计上位机刷写软件。整个上位机软件的设计遵循 V 型开发流程，首先提出设计需求，接着根据需求写出刷写方案，最后通过上、下位机联调测试，抓取报文，从而验证了本文设计的刷写软件的正确性及可靠性。

1 UDS 协议及帧类型介绍

1.1 UDS 协议

UDS 协议是诊断服务的标准规范，规定了诊断服务的具体命令，UDS 协议用于刷写的服务命令[4]见表 1。

表 1 UDS 协议服务命令

Tab. 1 UDS protocol service command

服务 Service	SID(0x)/h	诊断服务名
Diagnostic Session Control	10	诊断会话控制
ECU Reset	11	电控单元重置
Read Data By Identifier	22	通过 ID 读数据
Security Access	27	安全访问
Communication Control	28	通信控制
Rountine Control	31	例程控制
Request Rountine Result	33	请求例程结果
Request DownLoad	34	请求下载
Transfer Data	36	数据传输
Request Transfer Exit	37	请求退出传输
Tester Present	3E	通讯保持
Start Communication	81	开始通信

作者简介：唐恒飞(1994-)，男，硕士研究生，主要研究方向：汽车电子；王效金(1994-)，男，IOS 技术支持工程师，主要研究方向：汽车电子、智能数字锁。

收稿日期：2020-09-22

1.2　UDS 帧类型

UDS 协议可以应用于多种通信机制中，本文所研究的 UDS 协议是基于 CAN 总线的。因此 UDS 协议的消息帧是遵循 CAN 总线的报文格式。CAN 消息帧是由一个或者多个的数据协议单元（PDU）组成，而每一个 PDU 都包含控制命令（PCI）和数据信息（data）[5]。根据 PCI 和 DATA 的字节总数是否超过 8 个字节，可以将 UDS 协议帧分为单帧和多帧，其中多帧又包含首帧、连续帧、流控制帧。UDS 协议帧类型见表 2。UDS 协议帧类型可以通过前三个字节进行判别。第一个字节的高 4 位换算得到的值 0、1、2、3 决定了当前帧类别。在数据传输中，单帧相对简单点，遵循一问一答的机制，单帧中 SF_DL 代表着单帧传输中的数据字节数。然而多帧就比较复杂，多帧数据在传输时，上位机软件首先给控制器发送一个首帧数据，首帧数据中包含控制器即将接收数据字节的大小（*FF_DL*）；当控制器接收到该命令消息，给上位机回复一个流控制帧。流控制帧主要有 3 个参数，流控制帧状态（*FS*）、连续帧连续发送的最大数目（*BS*）以及 2 个连续帧之间的时间间隔（*ST_min*）。*FS* 有 3 个值，0、1 和 2 分别代表着继续发送、停止发送和数据溢出。通常在收到流控制帧时，该位常常是 0。对于 *BS*，主要对应着一定的范围（0 ~ 255）；当 2 个连续帧间的间隔时间超出 *ST_min* 时，ECU 端就会给上位机端回复一个发送错误的消息帧。最后就是不断地传输连续帧。连续帧中的 *SN* 代表着连续帧的连续号，每增加一个连续帧，*SN* 就加 1，直到增加到 15 时，*SN* 重新置 0。

表 2　帧类型

Tab. 2　Frame types

帧类型	Bit7-4	Bit3-0	Byte1	Byte2
单帧	*PCItype* = 0	SF_DL	N/A	N/A
首帧	*PCItype* = 1		FF_DL	N/A
连续帧	*PCItype* = 2	*SN*	N/A	N/A
流控制帧	*PCItype* = 3	*FS*	*BS*	*ST_min*

2　.hex 文件解析

上位机刷写的程序主要是用于 ECU 升级更新，这些程序经过一些软件编译而生成不同格式文件。.hex文件就是通过 Freescal CodeWarrior 软件编译而成的一种刷写文件。.hex 文件可以通过文本进行打开，打开后的文件内容是一行行记录。这一行行记录都是以冒号开头，内容是 16 进制码。.hex 文件格式见表 3。.hex 文件传输进 ECU 前，需要遵循 UDS 协议对 .hex 文件进行解析，提取记录数据长度、数据起始地址以及数据内容，提取完毕之后再转换成 CAN 报文格式将这些数据传输到 ECU 中。

表 3　.hex 文件格式

Tab. 3　.hex file format　/byte

冒号	本行数据长度	本行数据起始地址	数据类型	数据	校验码
字节	1	2	1	*n*	1

3　下位机设计

下位机是由实验室基于恩智浦公司的 imx6ull 芯片进行开发的汽车诊断仪，在此基础上利用 codeWarrior 软件编写 CAN 通信模块以及 UDS 协议栈，用于处理数据的接收与发送，整个设计图如图 1 所示。上位机通过串口将命令数据发送给诊断仪，经过 CAN 模块的初始化、接收、发送，通过 OBD 接口将数据发送给控制器；反之，控制器通过同样的流程将数据传送给 PC 端。

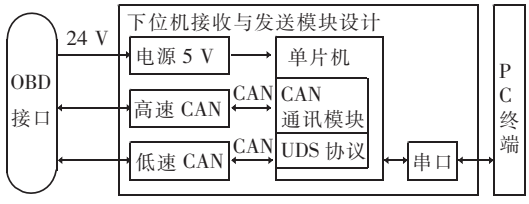

图 1　下位机设计图

Fig. 1　Design drawing of lower computer

4　上位机刷写软件设计

整个上位机刷写软件遵循软件开发中 V 型开发流程进行设计，包含设计需求、刷写流程、代码编写、软件测试、版本释放。本文主要研究的则是上位机软件设计的设计需求、刷写流程以及软件测试。

4.1　设计需求

根据某整车厂提供的控制器和 .hex 刷写文件，利用实验室研发的汽车诊断仪，设计一款上位机刷写软件。整个系统的架构如图 2 所示。

图 2　系统组成

Fig. 2　System composition

4.2　刷写方案设计

4.2.1　刷写流程

通过全面地研究 UDS 协议，整个刷写流程分为预编程阶段、主编程阶段以及后编程阶段[6]。对此可做阐释分述如下。

4.2.1.1 预编程阶段

预编程阶段是在进行主编程前的 CAN 网络准备。流程如图 3 所示。

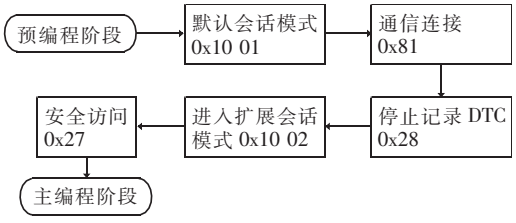

图 3 预编程阶段

Fig. 3 Pre programming stage

首先连通上下位机。通电之后,车载电控单元进入诊断默认会话模式[7]。上位机发送 0x81 请求与控制器进行通信,当上位机收到控制器回复一个包含 0xC1 的命令时代表通信成功。由于该软件的功能是刷写功能,上位机发送 0x28 命令,请求停止控制器对上位机进行诊断记录发送,这样可以让 CAN 总线不处于忙碌状态,在进行刷写时,能够大大减少刷写时间。车载电控单元的程序被更新时,需要电控单元的会话模式处于刷写模式下,因此发送 0x10 02 命令请求进入刷写模式。对控制器的 flash 区进行擦除和数据写入,是一个级别较高的操作,需要通过密钥才能进入[7]。向控制器发送 0x27 05 请求命令,控制器收到请求,给汽车诊断仪回复一个 0x67 05 aa bb cc dd 的消息帧,aa bb cc dd 代表 4 个 seekey,汽车诊断仪对这 4 个 seedkey 进行算法计算得到 4 个掩码,再发送 0x27 06 ee ff gg hh 给 ECU,ECU 拿到这 4 个掩码与内部的动态数据库进行匹配,匹配成功则会回复一个包含 0x67 06 的消息帧,代表允许上位机对控制器 Flash 区进行操作。

4.2.1.2 主编程阶段

主编程阶段主要分为 flash 区的擦除和数据写入。整个主编程阶段的流程图如图 4 所示。

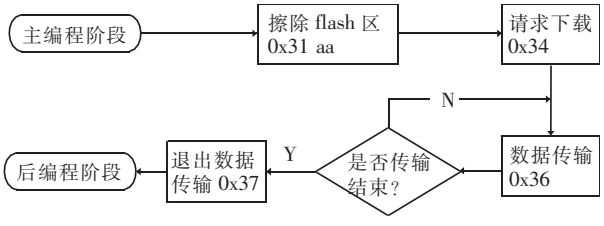

图 4 主编程阶段

Fig. 4 Main programming stage

主编程部分主要是分为 2 个部分:对 ECU 内部进行按地址擦除和数据写入[8]。擦除操作时,上位机发送 0x31 命令,通过解析. hex 文件,提取擦除的起始地址,以及擦除空间的大小,所以完整的擦除命令是 0x31 01 EE 00+xx xx xx xx(起始地址)+xx xx xx xx(擦除的空间大小)。当擦除完毕时,ECU 将会给上位机回复一个 0x71 01 EE 00 的报文消息,接着上位机软件与控制器确定需要下载的起始地址和长度,发送 0x34 命令,完整的报文是 0x34 00 44 +80 xx xx xx(起始地址)+xx xx xx xx(下载空间大小)。当 ECU 接收到该报文之后,将会给上位机回复一个含有 0x74 20 xx xx 的报文的正反应报文。已知程序传输的地址和最大长度之后,. hex 文件被解析得到的数据字节转换成 CAN 总线通讯格式传输到 ECU 中的固定地址区[8],上位机发送的命令是 0x36 01(传输的区块)xx xx xx xx xx...(传输的数据)。传输的过程是对 ECU 端的区块进行写入数据。当每传输一个区块的数据完成时,控制器将会给上位机软件回复一个含有 0x76 xx(当前传输的数据区块数)的报文,来表示当前该区块传输的数据成功。传输的数据量很大,因此需要多次地发送 0x36 命令,直到所有的数据传输完毕。当所有的数据传输完成,此时上位机软件将会发送 0x37 请求命令,请求退出数据传输。当该命令发出后,ECU 给上位机回复含有 0x77 的报文时,一个地址的刷写流程已经完整结束了,也代表着刷写中流程的结束。

4.2.1.3 后编程阶段

后编程阶段主要是校验写入的程序是否完整来判断整个刷写操作是否成功。后编程阶段的流程如图 5 所示。

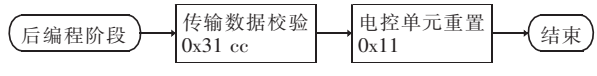

图 5 后编程阶段

Fig. 5 post programming stage

当控制器退出数据传输状态,上位机就立刻发送 0x31 01 DD FF 01 命令,请求检查刷写程序的完整性,收到 0x71 01 DD FF 01 命令就代表了此次车载电控单元程序刷写成功。上位机发送 0x11 命令请求,重新启动 ECU。

4.2.2 Winform 界面与数据库设计

Winform 界面主要有 3 个界面,即:登录界面、控制器选择界面以及刷写界面。界面设计用到一些简单的控件,如 label,button,textbox,ProgressBar 等。通过触发 button 控件的 *Click*( )事件,从而在界面上显示相关的信息以及进行刷写操作。数据库的设计是建立一张刷写文件的表格,关键字段有 ECU_Name,ECU_SoftWareNumber 和. Hex 文件名等。在读取控制的软件版本号,依靠这些关键字段的查找,

匹配成功之后从服务器后台下载刷写文件。

4.2.3　通讯逻辑层设计

本次设计的刷写软件属于自动化刷写，在选择好刷写文件之后，点击一键刷写，流程如图 6 所示。

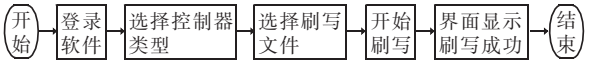

图 6　逻辑层流程图

Fig. 6　Logic layer flow chart

上位机软件抓取到的数据流都是通过串口进行发送和读取的，因此在逻辑层中设计了发送函数 *SendMeaasge*( ) 和读取函数 *ReceiveMessage*( )。客户端软件打开之后，通过 *ECUInit*( ) 函数和私有协议与汽车诊断仪连接成功。在选择控制类型时，触发 button 控件的 Click 事件，利用发送函数向串口发送包含 0x81、0x22 的命令用于连接控制器和读取控制器版本信息，将读取到的数据流进行解析，并带入数据库中去匹配，从而再从后台下载刷写文件。刷写文件直接存储到固定文件夹下，当点击刷写文件时，可以直接打开文件所在的位置。开始刷写是软件操作过程中最复杂的过程。. hex 文件传输进 ECU 前，需要遵循 UDS 协议对. hex 文件进行解析，提取记录数据长度、数据起始地址以及数据内容，提取完毕后再转换成 CAN 报文格式，将这些数据传输到 ECU 中。在 C#中，通过对文件流的读取，利用*substring*( ) 函数将所需的数据地址，数据内容等参数截取下来，存放到临时的字符串数组中，以便在数据传输时能够在线发送。

4.3　软件测试

为了验证刷写软件设计的正确性和适用性，进行了上下位机联调，上位机刷写界面和通过 CANList-2 抓取到部分的 CAN 报文信息分别如图 7、图 8 所示。从报文中可以看出，首先进入刷写模式下(0x10 02)，接着进入安全校验(0x27 05/06)，然后擦除对应的 flash 区(0x31 01 EE 00)，擦写完毕后，将需要写入的数据传输给控制器(0x36)，传输结束后将会退出传输状态(0x37)，为了检验刷写的数据是否完整，即发送请求进行校验(0x31 01 DD FF 01)，最后将控制器进行硬件重启(0x11 01)，刷写功能就完成了。从这些监测到的报文也就能够证明刷写的正确性和适用性。

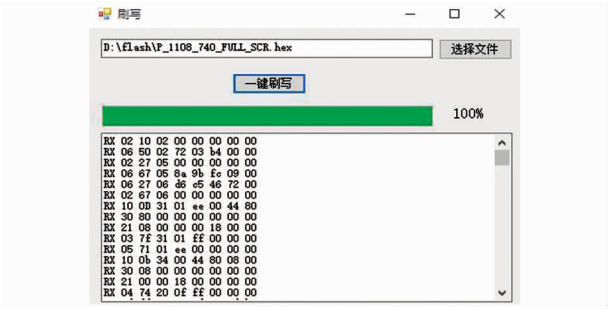

图 7　上位机 WinForm 界面

Fig. 7　Upper computer WinForm interface

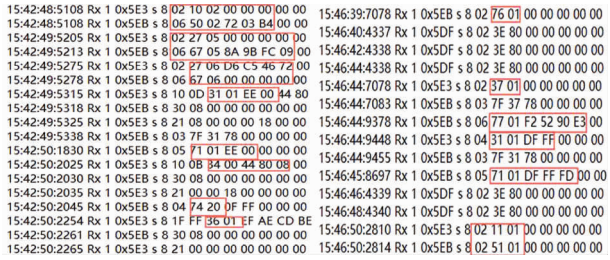

图 8　CAN 报文展示

Fig. 8　CAN message display

5　结束语

在本文中，利用 C#语言编写了上位机刷写软件，利用 V 型软件开发流程，为从事汽车控制器刷写的研究者提出了一种上位机刷写方法，而且基于 UDS 协议和 CAN 通讯机制对上、下位机之间的信息交互做了详细的介绍，也可为亟欲快速、全面地了解汽车控制刷写的初学者提供一些有借鉴意义的参考。但是本文的设计也存在着不足，诸如只是针对某整车厂的控制器进行刷写，未能对其他系统下的控制器进行刷写，因而亟待后续研究的进一步完善。

参考文献

[1] 马宏伟，吴长水. 基于统一诊断协议的控制器在线升级系统设计[J]. 软件工程，2020，23(8)：5-8.

[2] 李娇娇，张宏伟，陈金干. 基于 LabVIEW 新能源汽车控制器刷写软件设计[J]. 软件工程，2020，23(2)：16-18，8.

[3] 吴进军，方继根，王西峰，等. 基于 CAN 总线的新能源汽车 ECU 控制器程序刷写系统设计[J]. 机电产品开发与创新，2018，31(2)：1-3，7.

[4] ISO 14229. Road vehicles - unified diagnostic services [S]. Geneva，Switzerland：ISO，2012.

[5] 罗峰，孙泽昌. 汽车 CAN 总线系统原理、设计与应用[M]. 北京：电子工业出版社，2010.

[6] 王涛. 基于 CAN 诊断汽车控制器刷新软件的设计与实现[D]. 成都：电子科技大学，2015.

[7] 聂幸福，孟晨兴. 基于 UDS 的 BootLoader 上位机实现[J]. 汽车工业研究，2018(7)：26-29.

[8] 耿琦，葛亮，高东明，等. 基于 OTA 技术的车辆远程数据刷写研究及应用[J]. 电子测试，2017(15)：74-75.