Design and Implementation of Online Upgrade Software for Vehicle ECU Based on HIS Standard

Anyu Cheng [a], Liangbo Xiong [b], Ming Xie [c], Yang Li [d]

School of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

[a]chengay@cqupt.edu.cn, [b]xlb@sumarte.com, [c]xieming@sumarte.com,[d]haolyabc@126.com

Abstract

This paper has designed a flashloader software architecture which can be applied to different hardware platforms basing on hersteller initiative software (HIS) standard. This software architecture adopts hierarchical design thought and consists of five parts, which are micro controller abstraction layer, ECU abstraction layer, network layer, runtime environment (RTE) and application layer. The flashloader with this kind of structure can be transplanted conveniently and maintained easily. The software has integrated the diagnostic protocol stack which is based on the international diagnostic agreement, and solved the problem that current flashloader has low standardization of the agreement. Transplant the flashloader to different vehicle instruments which have different hardware platforms. After many times of experiments, the results show that this software can realize the online upgrade function with good performance, and be transplanted conveniently.

Keywords

Online Upgrade, Vehicle ECU, HIS standard, Flashloader.

1. Introduction

Now online upgrade technology of vehicle ECU has become more and more important. Due to the continuous development of micro controller technology, abundant software and hardware resources make it possible to realize the technology [1]. To the implementation of the technology, we can remove a single ECU from the vehicle firstly, and then use the dedicated download device to update the program [2]. But this way is very inconvenient and may damage the ECU easily. In order to solve the problems, Deng [3] proposed a method using serial port to achieve the online upgrade, but the method will increase the wiring harness, which may cause assemble and maintain the ECU will become more difficulty. Basing on this, Zhang and Tan [4, 5] proposed with the method using bus to realize online upgrade. But due to lack of protocol support, the accuracy and efficiency of the update data can't be guaranteed when in complex network environment.

At present, the automobile companies have developed the flashloader which suit for their agreement by defining their own protocol, ensuring the online upgrade progress can work normally [6, 7, 8]. However, the limitations of the software architecture and the disunity of the communication protocol causes that the current flashloader can't be reused and need to develop the whole software for different hardware platforms of ECUs. So in 2006, HIS organization published the complete flashloader norms, which standardized the diagnostic service layer and the transport layer which conform to the international diagnostic criteria, and defined the whole software architecture for Flash programing process [9]. Therefore, basing on the deeply analyzed of HIS standard, this article designed and implemented a flashloader which has good portability and versatility.

This paper is organized as follows: Section 2 introduces the main problems of current flashloader, and proposes a new flashloader software architecture based on HIS standard. Basing on this, Section 3 describes the implementation process of the software. Section 4 transplants and verifies the software on the different hardware platforms of instruments. Section 5 summarizes the full text.

2. Flashloader software architecture

2.1 Current problems of flashloader software architecture.

HIS standard specifies the implementation of software architecture for micro controller should adopt the hierarchical approach, and each layer use the standard function interface to call. But the current flashloader software architecture all defined by automobile companies [10, 11], which is not strictly according to HIS standard and dependent between each layers. For example, in the flashloader described in the papers which are presented by Zhang and Yan [12, 13], the protocol stack of network layer calls the functions of underlying driver layer directly, and operates the physical address of the Flash. Because in different ECU the physical address allocation of flash is different. The method may cause transplanting the upper layer software lack of unity, and need to redevelopment all the upper software for different ECUs, which may extend the development cycle and increase cost.

The implementation of current flashloader adopts the custom protocol mostly [15], such as the custom protocol is shown in Fig. 1 which is used in the paper by Liu.

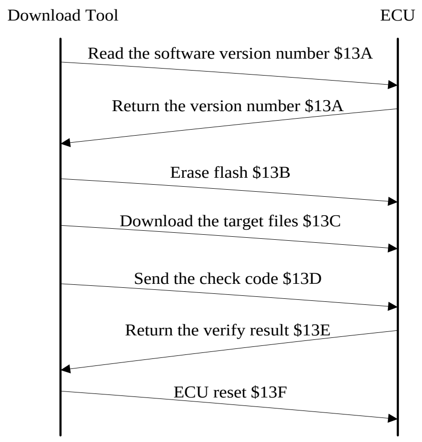

Download Tool
ECU
Read the software version number $13A
Return the version number $13A
Erase flash $13B
Download the target files $13C
Send the check code $13D
Return the verify result $13E
ECU reset $13F

Fig. 1 Custom protocol example

In the protocol, PC sends online upgrade command message to the target ECU to achieve online update function. But without the response mechanism for all request messages, the security and reliability of the data transmission are not guaranteed. In this protocol, the ID of the message is 0x13A - 0x13F, but in other custom protocols the ID may be different. The protocol of un-uniform is also the main factor of the difficult transplantation of the current online upgrade software.

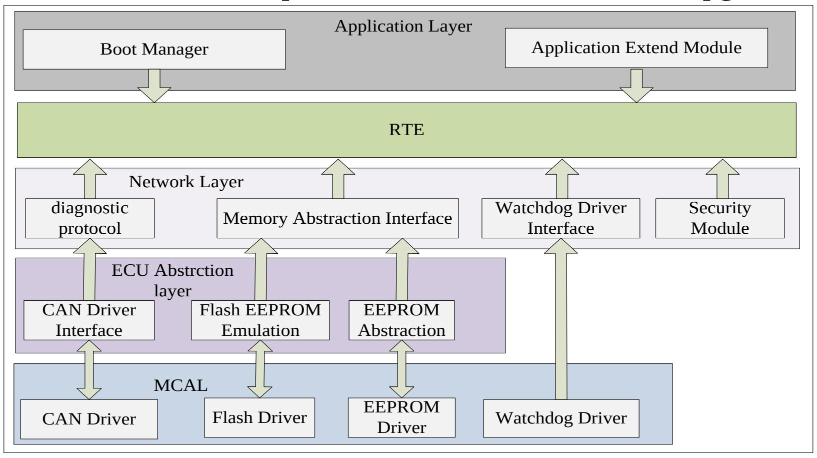


Fig. 2 Flashloader software architecture

2.2 Flashloader software architecture based on HIS.

In view of the problems of the present flashloader for vehicle ECU, after analysing the HIS software standard deeply, we adopt modular hierarchical design thought to design the flashloader software architecture, basing on the demand of the flashloader for ECU. The software architecture is shown in Fig. 2.

The software architecture contains five layers, which are micro controller abstraction layer, ECU abstraction layer, network layer, RTE and application layer. The micro controller abstraction layer connects the software and the actual micro controller, mapping the function of the micro controller and the peripheral interface. ECU abstraction layer is used to provide all driver programs on the basic of the hardware. Network layer provides various services for the software. The main work of the RTE is to handle the data exchange between the application layer and the other layers, which is the foundation of the software transplantable. The application layer is used to handle the business and logic work according to the actual demand.

The workflow of the software architecture is starting from application layer. Firstly, call the hardware initialization functions are stored in the micro controller hardware abstraction layer. Then run the boot manager module. The boot manager module writes and reads EEPROM through EEPROM driver interface, and judges the flag which is stored in EEPROM to decide whether the ECU is running application or flash loader currently. The detail work flow is shown in Fig. 3.

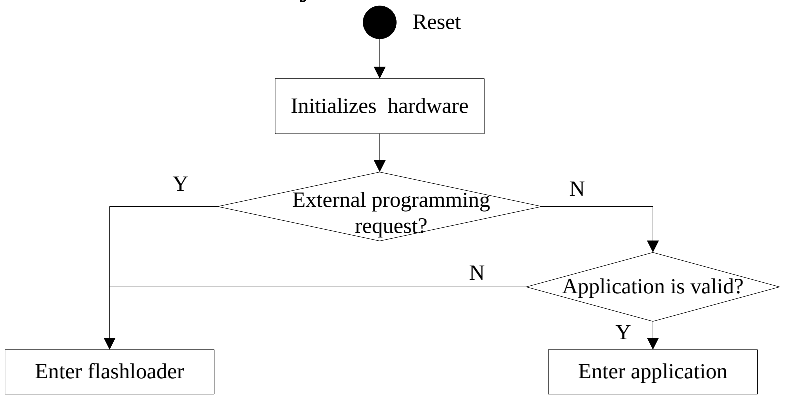


Fig. 3 Work flow of the boot manager module

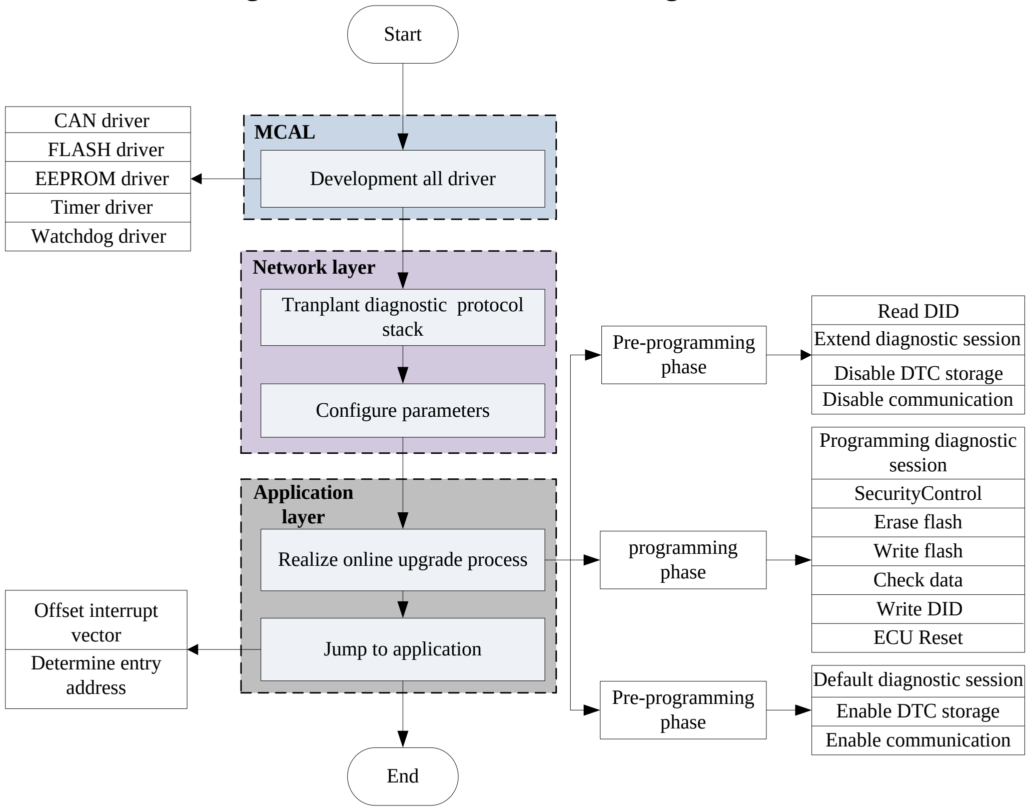


Fig. 4 The implementation process of flashloader

3. The realization of flashloader

3.1 Entire realization process.

When users are going to realize the software, first of all they should finish develop all drivers based on the specific ECU chip platform, including CAN, flash, EEPROM, timer and watchdog. Secondly, they should transplant the modular encapsulation source files to the network layer, which contains the network layer and the application layer of the diagnostic protocol stack, and configure all the configurable parameters. Finally, complete the whole process of online upgrade in the application layer, update and run the application. The detail implementation process is shown in Fig. 4.

3.2 Specific realization.

Due to each ECU has its own chip platform, all the drivers must be developed according to the specific chip platform. The realization of the diagnostic protocol stack is based on the international diagnostic agreement ISO 14229 and ISO 15765. To enhance the universality, the protocol stack specifies the format and the content of a message should adopt the international standards. For the convenience of transplanting the protocol stack in the different ECUs, this paper adopts the idea of modularization to encapsulate the diagnostic protocol stack, putting the data structure and the function prototypes of network layer and application layer in the header files, and then compiling the source files of functions to get the target code. Users just need to include the header files in their project and then compile all files and link the target code when operate the software.

The parameters which can be configured mainly include application layer parameters, network layer parameters, hardware platform parameters, message addressing mode, message ID and so on. The detail configuration information is shown in Table 1.

Table 1 The information of configurable parameters

Configurable parameter types	Parameters		Describe
Application layer parameters	services	SID 10, SID 11, SID 27, SID 22 SID 28, SID 2E, SID 31, SID 34 SID 36, SID 3E, SID 37, SID 85	Configure the services and its sub function according to demand
	Timer parameters	P2*Client P2*Server P2Client P2Server	Value scope:0-5000ms Value scope:0-2000ms Value scope:0-150ms Value scope:0-50ms
Network layer parameters	Control parameters	Stmin Blocksize	1ms 8byte
	Timer parameters	N_As, N_Bs, N_Cs N_Ar, N_Br, N_Cr	Value scope:0-150ms
Hardware platform parameters	Application layer	Main receive cache size Sub receive cache size Send cache size	Configure the parameters according to hardware resource
	Network layer	Network layer cache size	
Other parameters	Message addressing mode	Routine addressing Routine constant addressing Extend addressing Mixed addressing	Configure the message addressing mode and ID according to demand
	Message ID	Physical request message ID Function request message ID Response message ID	

The online upgrade process for vehicle ECU is the reprogramming process of flash, which mainly includes pre-programming stage, programming stage and post - programming stage. The main function of pre - programming stage is to check whether current programming conditions are suit for the demand of online upgrade for ECU or not, and configure the current network environment to ensure that ECU can enter into an appropriate programming environment. The main function of programming stage is to complete the external programming device access authorization, download flash driver and application, and then check all of the data. The main function of post - programming stage is to restore the normal communication and the DTC record of the other ECU on the network, and request to turn the current diagnostic session model into default session. The detail implementation steps are shown in Fig. 5.

After finishing update the application, programs running in the ECU should change from flashloader to application. Flashloader and application are stored in the flash memory with the method of partition. If the flashloader adopts the default interrupt vector table, the application should adopt the interrupt vector table which had been offset. The two interrupt addresses shouldn't be overlapped. Otherwise it will cause the interrupt conflict so that the application couldn't run normally. Due to different applications have different sizes, and their own application entry address, we must fix on the application entry address, insuring the ECU can jump to application normally.

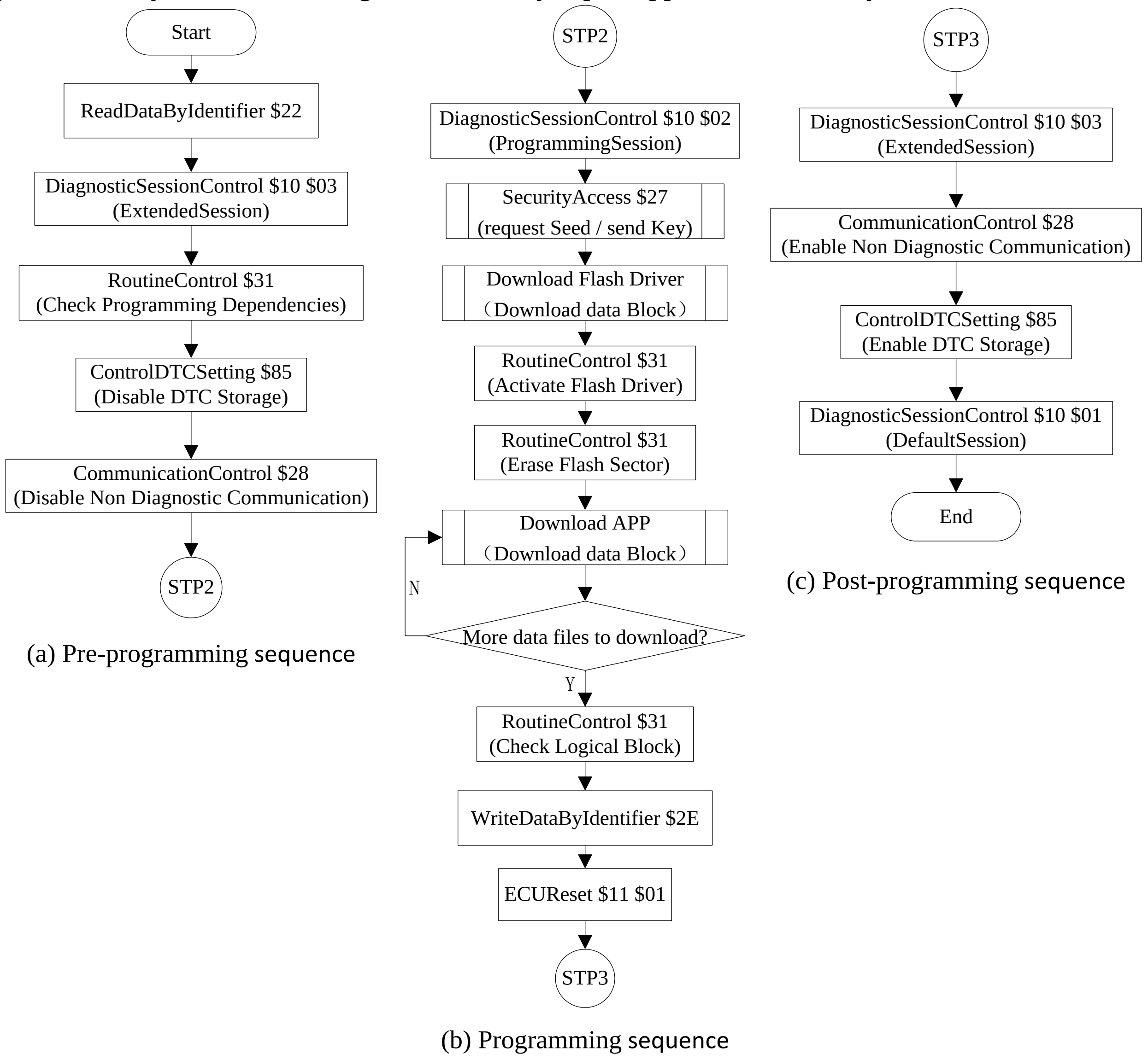


Fig. 5 Flash reprogramming process

4. Transplantation and verify the software

We transplanted the flashloader to two vehicle instruments and updated their application. The chip platforms of the two instruments are Freescale MPC5645S and TI TMS320F28035, both of which are 32-bit micro controllers, and their internal flash can be divided into different size of sections. Users can erase and write single or multiple sections assigned, and the detail memory allocation is shown in Fig. 6. To avoid happening the situation of erasing application by mistake, when the application is updated the flash driver is downloaded to the RAM by CAN bus. When power off the data of the flash driver will be lost, and not always stored in the flashloader programs.

We test the software on the two instruments. Firstly, we should connect the target ECU to the PC by USB - CAN adapter, and configure the detailed information of PC. Secondly, we should import the object files to the download tool and start to update the application. When the online upgrade process had been finished, the data which are stored in the target ECU flash memory as is shown in Fig. 7. The application image files of the two instruments as are shown in Fig. 8 and Fig. 9. By comparing the data we can find that all the data are exactly the same, which means that the target files have been written to the specified space accurately, and the online upgrade function is achieved.

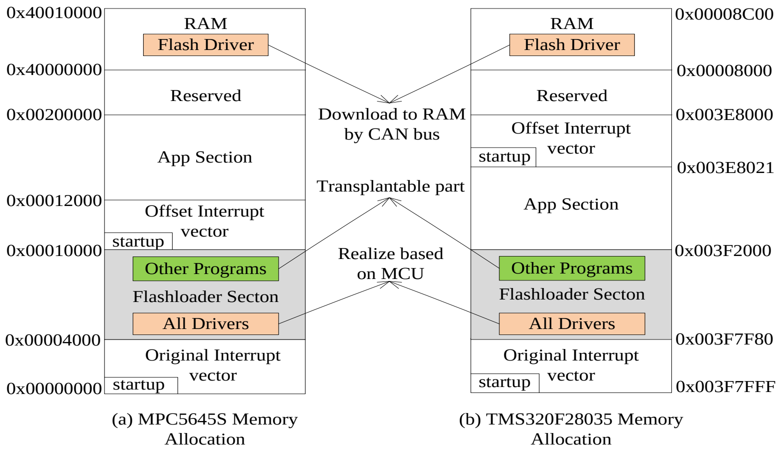
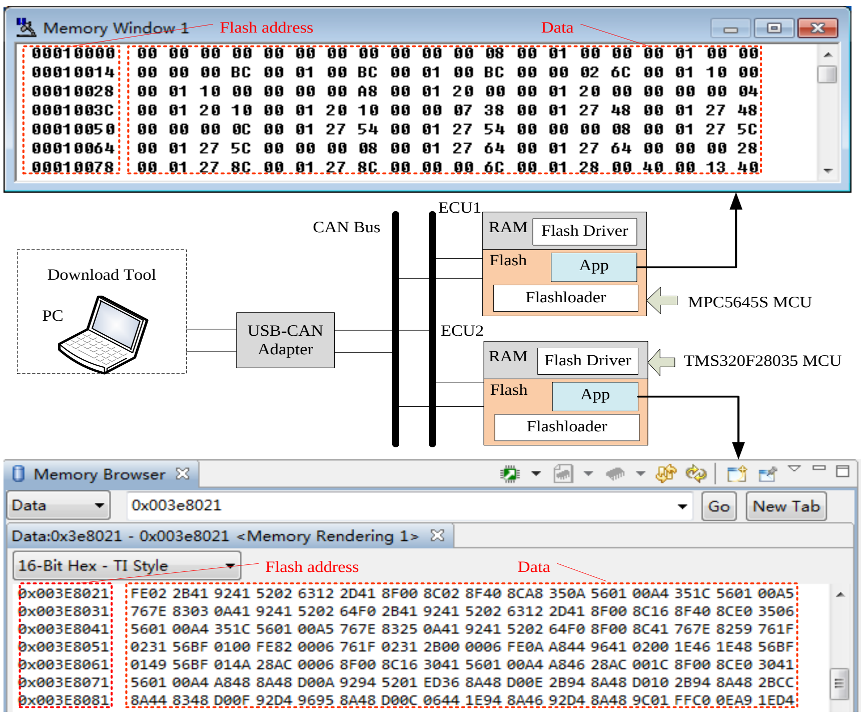


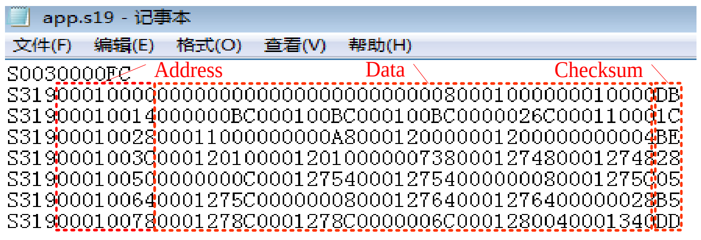
Fig. 6 Memory allocation map



```
00010000  00 00 00 00 00 00 00 00 00 00 00 08 00 01 00 00 00 01 00 00
00010014  00 00 00 BC 00 01 00 BC 00 01 00 BC 00 00 02 6C 00 01 10 00
00010028  00 01 10 00 00 00 00 A8 00 01 20 00 00 01 20 00 00 00 00 04
0001003C  00 01 20 10 00 01 20 10 00 00 07 38 00 01 27 48 00 01 27 48
00010050  00 00 00 0C 00 01 27 54 00 01 27 54 00 00 00 08 00 01 27 5C
00010064  00 01 27 5C 00 00 00 08 00 01 27 64 00 01 27 64 00 00 00 28
00010078  00 01 27 8C 00 01 27 8C 00 00 00 6C 00 01 28 00 40 00 13 40
```

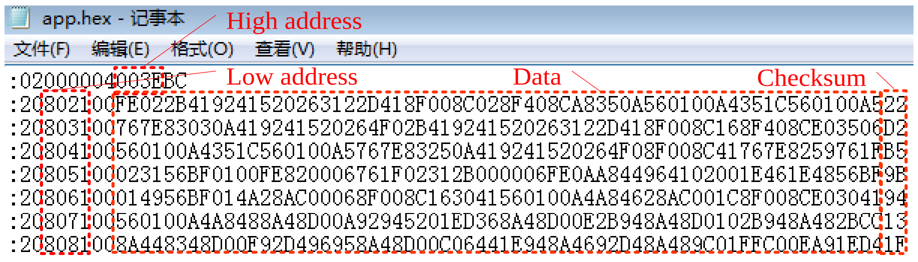
```
0x003E8021  FE02 2B41 9241 5202 6312 2D41 8F00 8C02 8F40 8CA8 350A 5601 00A4 351C 5601 00A5
0x003E8031  767E 8303 0A41 9241 5202 64F0 2B41 9241 5202 6312 2D41 8F00 8C16 8F40 8CE0 3506
0x003E8041  5601 00A4 351C 5601 00A5 767E 8325 0A41 9241 5202 64F0 8F00 8C41 767E 8259 761F
0x003E8051  0231 56BF 0100 FE82 0006 761F 0231 2B00 0006 FE0A A844 9641 0200 1E46 1E48 56BF
0x003E8061  0149 56BF 014A 28AC 0006 8F00 8C16 3041 5601 00A4 A846 28AC 001C 8F00 8CE0 3041
0x003E8071  5601 00A4 A848 8A48 D00A 9294 5201 ED36 8A48 D00E 2B94 8A48 D010 2B94 8A48 2BCC
0x003E8081  8A44 8348 D00F 92D4 9695 8A48 D00C 0644 1E94 8A46 92D4 8A48 9C01 FFC0 0EA9 1ED4
```

Fig. 7 The data of the target ECU



```
S0030000FC
S3190001000000000000000000000000000080001000000010000DB
S31900010014000000BC000100BC000100BC0000026C00011000 1C
S3190001002800011000000000A80001200000012000000000004BE
S3190001003C0001201000012010000000738000127480001274828
S31900010050000000C0001275400012754000000008000127505
S31900010064000127 5C00000008000127640001276400000028B5
S3190001007800012 78C0001278C0000006C0001280040001340DD
```

Fig. 8 Application image file of the MPC5645S platform



```
:020000040003EBC
:20802100FE022B4192415202631 22D418F008C028F408CA8350A560100A4351C560100A522
:20803100767E83030A41924152 0264F02B41924152026312 2D418F008C168F408CE03506D2
:20804100560100A4351C560100A5767E83250A41924152 0264F08F008C41767E8259761FB5
:20805100023156BF0100FE820006761F02312B000006FE0AA8449641 02001E461E4856BF9B
:208061000149 56BF014A28AC00068F008C1630415601 00A4A84628AC001C8F008CE0304194
:20807100560100A4A8488A48D00A92945201ED368A48D00E2B948A48D0102B948A482BCC13
:208081008A448348D00F92D496958A48D00C06441E948A4692D48A489C01FFC00EA91ED41F
```

Fig. 9 Application image file of the TMS320F28035 platform

5. Conclusion

This paper has designed and implemented a flashloader software architecture for vehicle ECU which can be applied to different hardware platforms. The software architecture adopts hierarchical design thought. The upper software calls the underlying drivers through HIS standard function interfaces, and ignores the details of the different hardware platforms. This software integrates the diagnostic protocol stack which is based on international diagnostic agreement ISO 14229 and ISO 15765, ruled the format and the content of the online upgrade message which should adopt the international standard. We have tested the flashloader on two different vehicle instruments in which the hardware platforms are different, and the result is that the software can be transplanted easily and can realize the online upgrade function normally.

References

[1] Yang J, Zhao G. The transplant and compile of Bootloader in embedded system[J]. Journal of Sichuan University, 2007, 44(4): 835-839.

[2] Zhang A Y, Zhu X M, Yang C. Development of the Bootloader Function of Electronic Control System[J]. Modern Vehicle Power, 2010, 4(4): 17-19.

[3] Deng H. Design of TMS320C6713 Flash Application Upgrading Online Based on the Serial Port[J]. Ship Electronic Engineering, 2009, 29(5): 120-122.

[4] Zhang H, Zhan D A. Design of Vehicle Fault Diagnosis System Based on CAN Bus[J]. Automotive Engineering, 2008, 30(10): 934-937.

[5] Tan T, Tang H, Zhou Y. Design and Implementation of Bootloader for Vehicle Control Unit Based on Can Bus[C]// Proceedings of the FISITA 2012 World Automotive Congress. Springer Berlin Heidelberg, 2013: 447-457.

[6] Mu C Y, Sun L N, Du Z J, et al. Research and Development of Device for Downloading and Updating Software of Product ECU Based on Extended CCP[C]//2nd IEEE Conference on Industrial Electronics and Applications. IEEE, 2007: 2865-2869.

[7] I Zhang Y, Bao K J. Design and Implementation of BootLoader for Vehicle Control Unit[J]. Computer Engineering, 2011, 37(12): 233-235.

[8] Wang Y, Lin Z. Stable In-circuit Programming of Flash Memory in Freescale's MC9S12 MCU Family[C]// Proceedings of the 2010 International Conference on Measuring Technology and Mechatronics Automation - Volume 03 IEEE Computer Society, 2010: 477-480.

[9] DaimlerChrysler AG. HIS Flashloader Specification Version 1.1[S]. 2006.

[10] Cheng A, Yao Y, Duan Z, et al. ECU loader design of in-vehicle CAN network based on ISO15765[C]// Information Science and Technology (ICIST), 2011 International Conference on IEEE, 2011: 1215-1217.
[11] Fahrner D I A, Geese D I M, Happel D I A. Data compression algorithms for EOL flash programming of automotive electronic control units[J]. Atzelektronik Worldwide, 2010, 5(2): 22-26.
[12] Zhang D, Wu Y. The research and implementation of flash EMIFA bootloader[C]// Image and Signal Processing (CISP), 2012 5th International Congress on IEEE, 2012: 1911-1914.
[13] Yan J. Design of Secondary Bootloader for Embedded System based on DSP [J]. International Journal of Computer Science Issues, 2013, 10(1): 544-548.
[14] Xu Y, Wang R G, Cheng A Y, et al. Design of Online Upgrade System for the Software of Vehicle ECU based on CAN-Bus[J]. International Journal of Advancements in Computing Technology, 2013, 5(1): 79-87.
[15] Liu C, Luo F. Implementation of CAN Bootloader Based on Freescale MCU [J]. Journal of Suzhou University, 2010, 30(2): 57-61.