LIN Flash Bootloader Based on UDS

Feng Luo and Yue Yin Xie
Clean Energy Automotive Engineering Center, School of Automotive Study, Tongji University, Shanghai, China
Email: luo_feng@tongji.edu.cn, xieyy135@126.com

***Abstract*—Flash Bootloader is necessary for updating or maintenance of the ECUs on the Vehicle. This paper discusses how to develop the LIN flash bootloader based on Unified diagnostic services. First is to discuss the software structure of the flash bootloader, the next is to discuss the details of hardware layer, LIN driver layer and UDS layer of the structure, and then is to discuss the solution for reliability. After all these works, test the software with a standard diagnostic tool. The result proofs that the LIN flash bootloader works properly as designation, the software follows to the LIN protocol and Unified Diagnostic Services protocol strictly, and the ECU can restore from the errors during downloading period.**

***Index Terms*—flash bootloader, LIN, unified diagnostic services, reliability**

I. Introduction

More and more electronic control units (ECUs) are used on the vehicle. Updating the program of the ECUs becomes an important issue for the OEM. It needs an easy and convenient solution to updating the program of ECUs. Flash bootloader is necessary for the ECUs.

The FBL for the vehicle ECUs shall translate data via CAN or LIN Bus. Currently, the two most widely used buses on the vehicle are CAN and LIN. Compared with CAN, LIN is based on the normal UART controller, which exists in nearly all the MCU. The data link layer of LIN is realized by software and doesn't need the specially designed LIN controller. That can save a lot of money for the OEMs. The speed of LIN Bus is up to 20kbps, which is suitable for the controllers of window, door and sunroof, sunshade etc. And this paper selected LIN Bus as the communication solution.

The FBL shall also need to be executed according to a standard protocol. Road Vehicle-Unified Diagnostic Services according to the ISO14229 is the standard diagnostic protocol for automotive industry, so this paper select UDS as the diagnostic services.

The following chapters will discuss how to design the flash bootloader. And a MCU named MC9S12XEP100 from Freescale Company is selected as the target hardware to test the result of the FBL program.

II. System Structure

LIN Flash bootloader based on UDS contains 4 layers: Hardware layer, LIN Driver layer, Unified Diagnostic Services Layer and Application layer, as Fig. 1 shows.

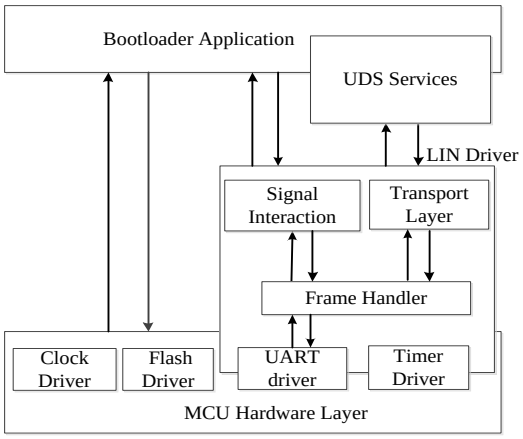

Figure 1. Structure of LIN UDS flash bootloader.

Hardware layer works as the basic drivers of the hardware. It is better for the bootloader to have smaller code size, so hardware layer only need to realize MCU clock driver, UART driver, timer driver and Flash driver.

LIN driver layer bases on the UART driver and timer driver, frame handler is responsible for all the LIN data link layer, the signal interaction module translates the data between the real application signal and LIN data frame. LIN driver has integrated the transfer layer internal, which is used for translating one or more frames of LIN Bus.

The Unified Diagnostic Services Layer according to the ISO14229 is responsible for the diagnostic controls. The UDS for LIN bases on the translate module and diagnostic frames of LIN driver.

Application layer can access to the hardware layer directory, such as initial the clock system, erase, write and read flash. It can also interacted with LIN driver directly, all the frames except diagnostic is interacted between application layer and LIN driver directly. For Flash bootloader, the application layer gets the UDS requests from the tester, and calls the hardware driver functions directly. After finished the work, the application layer translate the result by UDS response frame to the tester.

This paper mainly discuss about Hardware layer, LIN driver layer and UDS layer.

Manuscript received December 11, 2014; revised March 11, 2015.

doi: 10.12720/joace.4.1.47-52

III. HARDWARE DRIVER LAYER

The target hardware is MC9S12XEP100. This paper don't discuss the basic driver of clock, UART or flash, but focus to the locating of interrupt vector table and protect of flash driver, which are import for flash bootloader.

A. Locating of Interrupt Vector Table

The flow of a hardware interrupt routine is as Fig. 2. When detect the trigger of the external interrupt, check the interrupt vector table, get the interrupt service routine address, jump to this address to execute this interrupt routine, and after finishing this routine, return the main routine.

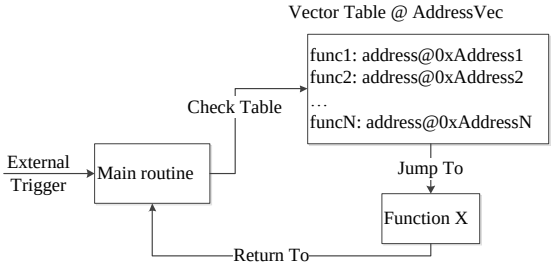

Figure 2. Flow of interrupt service routine.

For the application without flash bootloader, there is only one vector table, and the compiler will locate this table to the default address. Take XEP100 for example, the compiler locate this table to the default address 0xFF00 [8]. As Fig. 3 shows.

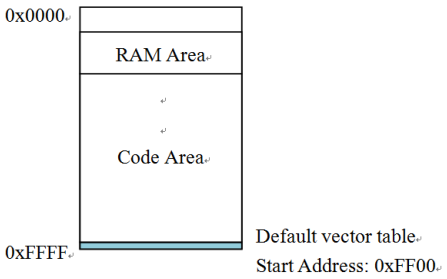

Figure 3. Default interrupt vector table.

For the application integrated with flash bootloader, there are two interrupt tables existing. One is for Flash bootloader and the other is for the application [6]. In this paper, the interrupt table located at the default address is assigned to flash bootloader and another interrupt table located at 0xBF00 is assigned to application. As Fig. 4 shows.

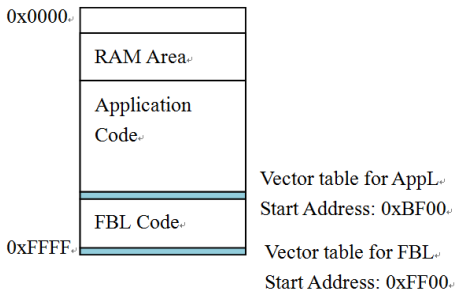

Figure 4. Two interrupt vector table.

B. Security of Flash Driver

Flash driver contains the erasing, reading, writing of Flash. This chapter doesn't discuss the detail but the security of flash driver.

For the ECU, it is very important to protect the program internal from modifying or damaging unexpectedly. So, there shall be security of flash driver to ensure they are called intentionally. This paper uses the solution called relocating program address to ensure the driver are not executed unexpectedly, even if the program counter of CPU points to the flash function entry address saved in the ROM.

Relocating program means relocate the real run address of the program to the address in the RAM instead of the address which saves the codes in the ROM. When the function is needed to be called, the ECU copies this function from the ROM address to the relocated address in the RAM. And then jump to the function entry address in the RAM. After finishing the operation, clear the area which saves this function in the RAM, and return to the main routine. Take XEP100 for example, to relocate the code by modifying the linker file as following:

SAMPLE_CODE_SEG = READ_ONLY 0xFD00 TO 0xFEFF RELOCATE_TO 0x3D00.

These codes notify the compiler to compile the code saved between 0xFD00 to 0xFEFF area using the run address between 0x3D00-0x3DFF. Fig. 5 shows how to call the relocated functions:

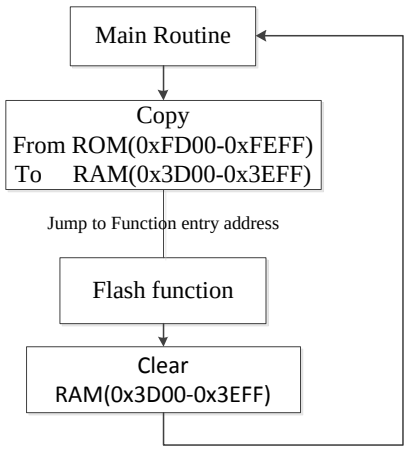

Figure 5. Flow of call flash driver.

IV. LIN DRIVER LAYER

This paper discusses the transport control layer and diagnostic frames of LIN other than the detail of LIN bus, which have connection with the LIN Flash Bootloader.

A. Transport Control Layer of LIN

LIN Bus usually works as the sub-bus of CAN BUS. The tester is not connected to LIN Bus directly. Diagnostic is executed via the CAN Bus and integrated with LIN Bus via CAN/LIN gateway. As Fig. 6 shows. Transport control layer of LIN of Bus is used to have the same diagnostic capabilities as CAN Bus.

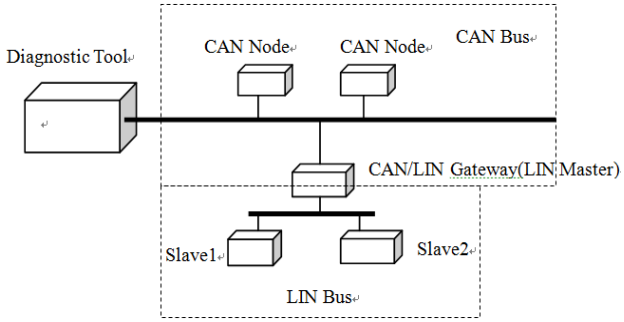

Figure 6. Typical set up for LIN Bus diagnostic.

To translate data containing one or more frames, some translate control information should add into the LIN data frame. The LIN frame with transport control information is called PDU (package data unit). Messages issued by the client are called request and messages issued by the server are called response. The structure of the PDU is as Fig. 7 [2].

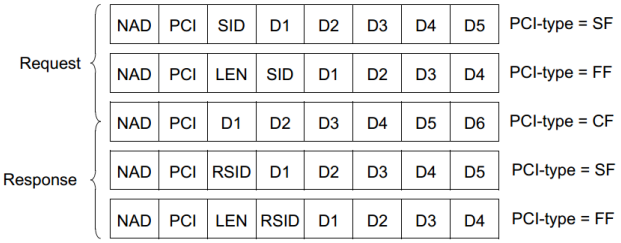

Figure 7. PDU structure.

The first byte contains the NAD (Node Address for diagnostics) by which a LIN slave is addressed in the case of a master request frame. Only a LIN slave that has already been addressed by a master request frame is authorized to use the diagnostic response.

The second byte contains the PCI (Protocol Control Information), which specifies the type of diagnostic frame: according to PCI, transported frame can be identified as a complete message (Single Frame, SF) or a part of the message (First Frame, FF or Consecutive Frame, CF).

The third byte contains the SID (or RSID) that specifies the request (or the response) shall be performed by the slave node addressed. The interpretation of the subsequent data bytes for these messages is provided in this diagnostic specification and depends on the SID or RSID (see Chapter 5 about diagnostic services).

B. *Diagnostic Frames of LIN*

There are two identifier specially designed for diagnostic frames. ID 0x30 is used as master request frame and ID 0x3D is used for slave response frame [2]. Take the bus system as Fingure6 for example. The tester send the diagnostic request, this frame is translated to CAN/LIN gateway. Then the gateway which works as client translates the request frame with 0x3C identifier to the slave node, which works as server. And the slave node will query the diagnostic module whether it shall response or silent. If it shall response for the request, it will translate the diagnostic frame with 0x3D identifier. During the response period, the master shall send the response frame header unconditionally. The work flow as Fig. 8:

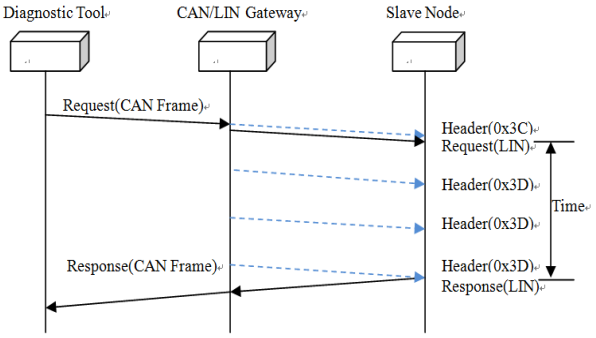

Figure 8. Diagnostic frame flow.

The time value in Fig. 8 is specified in LIN protocol. It is minimal value is 50ms, which defines the time between reception of the last diagnostic request frame on the LIN bus and the slave node being able to provide data for a response. And the maximum value is 500ms, which defines the time after which a slave node must receive a slave response header before it discards its response.

V. Unified Diagnostic Services

A. *Introduction of UDS*

Unified Diagnostic Services (UDS) are established in order to define common requirements for diagnostic system, whatever the serial data link is [1]. It bases on the open systems interconnection (OSI) basic reference model in accordance with ISO7498-1 and ISO/IEC 10731, which structures communication systems into seven layers. When mapped on this model, the services used by a diagnostic tester (client) and an ECU (server) are broken into: unified diagnostic services as layer 7 and communication services (layer1-6) [5].

For the ECU with LIN communication, the map of 7 layer module is as Table I:

TABLE I. Seven Layer Module of ECU with LIN Bus

Application Layer	ISO14229(UDS)
Present Layer	--
Session	LIN Driver
Transport	LIN Driver
Network	LIN Driver
Data Link	LIN Driver
Physical	LIN Transceiver

B. *Work Flow of the FBL Based on UDS*

Only a part of services of UDS are needed for FBL. For the ECU integrated with UDS, the reprogramming steps are as Fig. 9. These steps can be divided into 4 parts: preparation for reprogramming of ECU, Security access, jump to the FBL and download program in FBL.

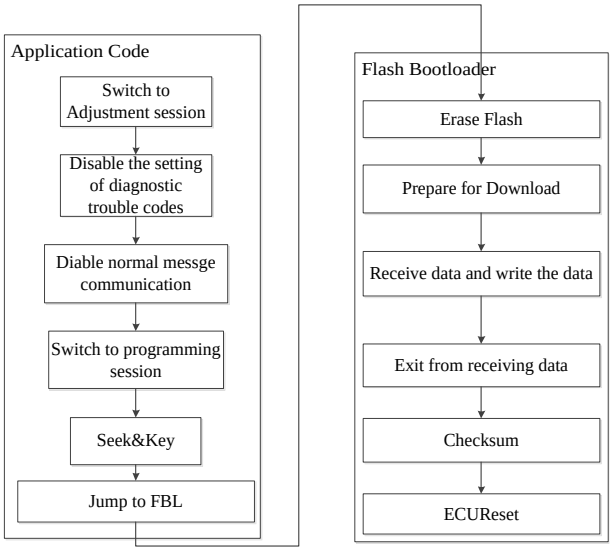

Figure 9. Reprogramming step of ECU.

C. Preparation for FBL

Before get the reprogramming request, the ECU is running in the application mode. Some preparation is necessary before jump to the FBL mode. It contains the following 4 steps to disable some function module and switch to the programming mode: use session control service (SID = 0x10) with value 0x03 to switch the ECU to adjustment session; using control DTC setting service(SID = 0x85) to disable the setting of diagnostic trouble codes, which may cause unintended detection of fault conditions during download; using communicating control service(SID = 0x28) to disable the translating and receiving of normal messages, which are not needed for flash bootloader; using session control service with value 0x02 to switch back to programming session.

Take session control as example to explain how the ECU interacted with the diagnostic tools based on UDS.

First, Diagnostic tools send the session control request. The data package of the request is [1]:

Request	0x10	0x02:programming session	

ECU gets the request and switch to programming session. If this operation is success, then send the positive response, the data package is [1]:

Positive	0x50	0x02:programming session	

If the operation fails, then send the positive response to the diagnostic tool, the data package is [1]:

Negative	0x7F	0x10: session control ID	EC

D. Security Access

The program and data saved in the ECU concern the safety of the vehicle. ECU uses seed and key mechanism to forbid the illegal access to the ECU. Fig. 10 shows the work flow of seed&key.

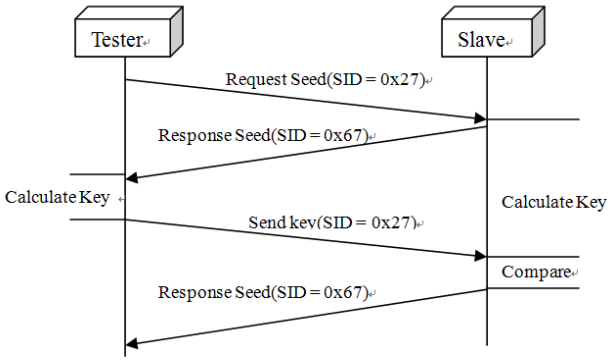

Figure 10. Work flow of seed&key.

The diagnostic tool send the request (SID = 0x27) with value 0x01 to request the seed from the ECU. Seed value is a random number generated by ECU. ECU translate the response (SID = 0x67) with the seed value to the diagnostic tool. The tool calculates a key with the seed and private key and translates the key to the ECU. ECU checks whether the key from the tool matches to the key calculated itself. If it matches, the diagnostic tool gets the authority to access the ECU.

E. Jump to the FBL

After get the authority to access ECU, the diagnostic tool send routine control of jumping to FBL request to the ECU. Then the ECU sets a global variable equals to macro definition as FBL_MODE, which is saved in the RAM and will not be changed during the reset. After startup, the first step is to check this global variable in the RAM. If this variable equals to FBL_MODE, jump to the Flash bootloader program. Fig. 13 shows the work flow of jumping to the FBL program.

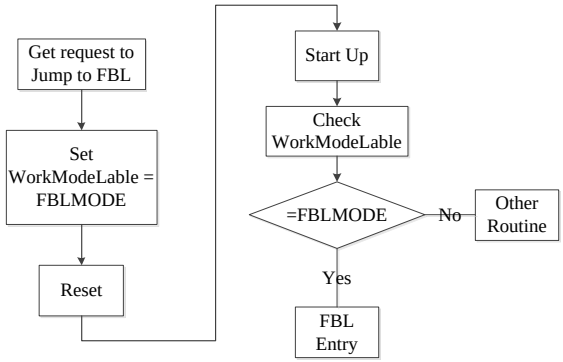

Figure 11. Work flow of jumping to FBL program.

F. Download Program

Downloading program with flash bootloader mainly contains the following steps: using routine control of erase service to erase all the flash sectors which are used to save the application code; using request download service(SID = 0x34) to notify the ECU prepare to download data; using translate data service to translate program data to the ECU; using translate exit service to notify the ECU exit the receiving data mode; using routing control of checksum service to check correctness

and completeness of the reprogrammed codes; using ECU reset service to reset the ECU and restart to run into the application program.

It takes a long time to execute the routine control of erase and checksum services. The main routine of ECU shall not wait until finished erasing flash or calculating checksum, otherwise the ECU will miss all the LIN communication during this period. In this paper, the solution is to create a sub-routine control function. When the UDS module receives the request to erase flash or checksum, set a flag to notify the sub-routine control function to start erasing or checksum. The sub-routing control function is running in the main loop and process part of the work every cycle. After finishing all the work, the sub-routing control function will notify the UDS module that result is ready. And then, the UDS module translates the result to the diagnostic tool.

VI. Reliability of FBL

A. Error Handling

If there are some unexpected errors happen during download program. The diagnostic will try to send the request for 5 times [4]. If the ECU restores from the errors in the 5 times and response to the diagnostic tools successfully, FBL works continually, otherwise the ECU will be reset to wait for another work flow of FBL.

If there are terrible errors happen during downloading and it makes the program of the ECU blocked. The hardware watchdog will monitor this situation and reset the ECU.

B. Integrity of Program

LIN FBL uses checksum to ensure that it downloads the program to the ECU without losing or changing data. It mainly contains two parts of work: calculating and integrating the checksum of hex file; checksum service of ECU.

For the first part, it needs PC software that can calculate the checksum and integrate the value to the hex file. Fig. 12 is a sample work flow of processing the hex file. In this paper, the flash area addressed at 0xBF00 is used to store the checksum value. The checksum is downloaded to the ECU as other codes in the hex file.

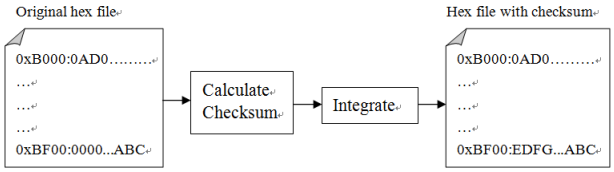

Figure 12. Calculate and integrate checksum of hex file.

After finishing downloading program to ECU, the diagnostic tool will send the checksum service request to the ECU. Then the ECU calculates the checksum with the value read from the flash area, and compare it with the one stored at 0xBF00. If matches, it means the hex file has be downloaded to the flash without lost or changing.

VII. Test of LIN FBL

This paper uses the LIN-flashing tool software from IHR GmbH to test the LIN FBL software. IHR GmbH is a German vehicle electronic company and is one of the 3 testing centers of LIN BUS certified by Volkswagen.

A. Test with Correct Private Key and Hex File

The first step is to test the software with correct private key and hex file to check whether the LIN FBL can work properly with the standard diagnostic tools. As Fig. 13 shows, the program is downloaded to the ECU smoothly. That means the LIN Flash Bootloader designed in this paper follows the LIN protocol and Unified Diagnostic Services strictly.

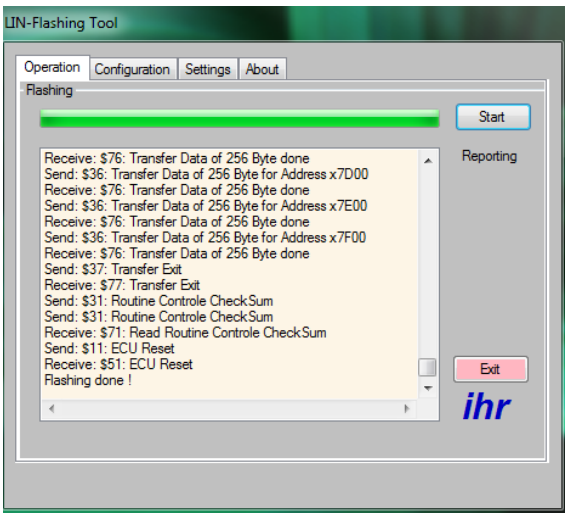

Figure 13. LIN Flashing Tool downloads the program to the MCU successfully.

B. Test with Wrong Private Key

The second step is to test with wrong private key to check whether the LIN FBL can protect the ECU from illegal access. As Fig. 14 shows, the illegal access is denied by the ECU. The diagnostic tool will try to send request for 5 times if the last one fails. After that, reset the ECU to wait for another work flow of bootloader.

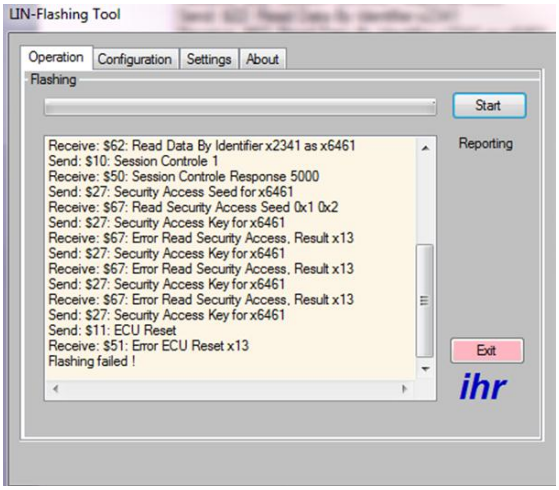

Figure 14. The illegal access is denied by the ECU.

C. Test with Corrupted Hex File

The third step is to test with corrupted hex file to check whether the LIN FBL can detect the fault of changing or losing data during downloading. As Fig. 15 shows, the FBL detect the fault of checksum successfully. After 5 times retries, The ECU is reset to wait for the next work flow.

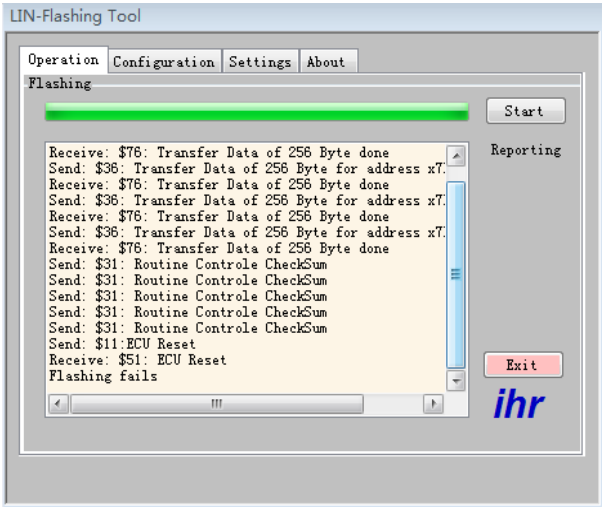

Figure 15. Downloading fails with corrupted hex file.

VIII. Conclusion:

The paper supplies a solution of FBL to the vehicle with LIN nodes. With the FBL, the tester only needs to connect the tool to the standard interface of the vehicle and update the program of the ECU, which takes great convenience to the tester.

The FBL bases on Unified Diagnostic Services. The UDS is an open protocol and all the related company can develop its own UDS software package and diagnostic tools according to the protocol documents. That makes the OEM have more choices to select the best solution for the ECU. And the UDS is designed according to the open systems interconnection (OSI) basic reference model, it is benefit for the vehicle to be integrated into the internet of things in the future.

References

[1] Road vehicle–Unified diagnostic services (UDS)–specification and requirements.
[2] LIN Specification Package Revision 2.1. LIN-SUBBUS, 2006.12.24.
[3] Salcianu, M. Fosalau, and Cristian, "A new CAN diagnostic fault simulator based on UDS protocol," in *Proc. International Conference and Exposition on Electrical and Power Engineering*, Iasi (RO) 2012.
[4] T. Liu, Y. H. Zhao, W. J. Cai, W. Q. Chen, X. M. Wei, and F. Q. Zhao, "Troubleshooting mechanism and strategies of automotive communication based on UDS," *Vehicle Electric*, no. 2, 2013.
[5] Y. Zheng, F. F. Li, and F. Luo. "Vehicle remote diagnostic system implementation based on 3G communication and browser/server structure," in *Proc. 2012 International Conference on Information Engineering*, Singapore, 2012.
[6] M. W. Xue, Z. M. Ri, and C. Xiaohui. "Design and realization of bootloader in embedded system," *Computer Engineering*, vol. 31, no. 7, 2005.
[7] W. Yagang. "Analysis and transplant of embedded bootloader mechanism," *Computer Engineering*, vol. 36, ch. 6, 2010.
[8] *MC9S12XEP100RMV1*, Freescale Semiconductor. 2010.09

Feng Luo Feng Luo received his PhD degree in Aircraft Control Engineering from Northwestern Polytechnical University in 2000. From 2000 to 2002, he did his postdoctoral research work in the Department of Automotive Engineering of Tsinghua University. Feng Luo came to Tongji University in 2002, he is now a Professor in Automotive Engineering Study of Tongji University. His research interests include: Automotive Networks, Automotive Electronic System Control , Vehicle to Vehicle Communication.

Yueyin Xie Yueyin Xie starts his Postgraduate in Tongji University from 2010. During this time, his main research includes: Vehicle CAN Networks, Vehicle to Vehicle Communication and Data Recorder. Now he is doing his PhD program at Tongji University. The research for PHD is Conformance test of AutoSAR.