Frontiers in Science and Engineering Volume 5 Issue 2, 2025
ISSN: 2710-0588

Design of Automotive Software Upgrade Simulation Test Bench

Yongjian Zhu[1, 2, a], Ying Jing[3, b], Manna Wang[1, 2, c], Yuzhe Wang[1, 2, d], Kaiquan Liu[1, 2, e]

[1]China Automotive Inspection Center (Tianjin) Co., Ltd., China

[2]China Automotive Technology Research Center Co., Ltd., China

[3]China Quality Certification Center Co., Ltd., China

[a]zhuyongjian@catarc.ac.cn, [b]690966326@qq.com, [c]wangmanna@catarc.ac.cn, [d]wangyuzhe@catarc.ac.cn, [e]liukaiquan@catarc.ac.cn

Abstract

With the rapid development of automotive electronics technology, automotive software upgrades have become an indispensable part of the modern automotive industry. To ensure the safety and reliability of software upgrades, the design of simulation test benches has become particularly important. This article aims to explore a design scheme for an automotive software upgrade simulation test bench. Through a detailed analysis of the hardware and software architecture of the bench, an integrated testing platform is proposed. This platform can simulate real car operating environments and conduct comprehensive testing of software upgrade processes, including functional verification, performance testing, compatibility testing, etc. The effectiveness and practicality of the test bench have been verified through practical application cases. This study provides a new solution for automotive software upgrade testing, which helps to improve the quality and efficiency of automotive software upgrades.

Keywords

Software upgrade; Bench simulation; Testing; OTA.

1. Introduction

With the rapid development of automotive electronics technology, modern cars have become a highly integrated and intelligent system. As a core component of automotive intelligence, the quality and safety of automotive software are directly related to the driving safety and user experience of the car. However, with the continuous increase in software functionality and complexity, software upgrades have become an inevitable part of the automotive industry. To ensure the safety and reliability of software upgrades, the design of simulation test benches has become particularly important. In the OTA simulation testing system, the universal bench simulation module is a key component that allows testing teams to simulate and validate the OTA upgrade process outside of the real vehicle environment. By using a development board to simulate ECU, testers can build a multi communication protocol simulation platform to simulate the vehicle network communication environment and ECU functional behavior. The main purpose of this simulation platform is to connect with the OTA cloud platform, simulate the real OTA cloud management end link function, so that the testing team can obtain the upgrade package and verify its integrity, effectiveness, and correctness. The testing team can develop a series of test scripts to verify various aspects of vehicle cloud communication and in vehicle communication. These test scripts can cover multiple aspects such as cloud interface testing, human-computer interaction testing, process testing, stress testing, etc., to ensure that every step of the OTA upgrade process meets the expected requirements. The simulation test bench can simulate the real automotive operating environment and conduct comprehensive testing of the software upgrade process, including functional verification, performance testing, compatibility testing, etc. Through simulation testing, potential issues can be identified before software upgrades, reducing upgrade risks and improving the quality and efficiency of software upgrades. Therefore, this article will explore a design scheme for an automotive software upgrade simulation testing bench, in order to provide a new solution for automotive software upgrade testing.

2. Design Concept for One Rack

Based on the emphasis on online concurrency, system architecture flexibility, and scalability in the process of transmitting information through large-scale vehicle networking and obtaining OTA upgrade packages, our OTA management system core emphasizes lightweight, fast response, and rapid expansion, and has cross platform capabilities.

It can be quickly deployed on multiple platforms and is easy to establish a high availability architecture. The database uses MongoDB, which has the characteristics of elasticity and fast query response. Developed by nodejs&Java, this backend access RESTful API supports HTTPS. It can query new versions of software and obtain upgrade task parameters through JSON objects. Each upgrade task can include one or more target devices to be upgraded. Each terminal ECU can be upgraded using a full package or incremental upgrade, with the upgrade package URL, The upgrade sequence, software version dependency, current software version number, new software version number, existing version backup path, upgrade software digital signature, and whether the upgrade failure is rolled back as a whole for multiple modules can be set. The device side can obtain the upgrade package for the upgrade task setting, verify the digital signature, and check the version number, device information, dependent function information, etc. of the target device according to the set sequence. After meeting the settings, it enters the upgrade setting process, backs up the existing software to the set path, upgrades the target device, and notifies the next device according to the upgrade setting process after completion.

If the upgrade fails, the backup path will be flushed back to the previous version before the upgrade. If it is a multi module upgrade, check whether the upgrade settings need to be rolled back as a whole in case of failure. If so, notify other devices to flash back to the previous version. After processing, report the processing process to the backend, The OTA backend can provide upgrade reports for visual tracking. The OTA management platform can set the upgrade sequence, software version dependency, current software version number, new software version number, existing version backup path, upgrade software digital signature, and whether to roll back the entire upgrade failure for multiple modules. And it can reserve data exchange interfaces with TSP, PKI/CA systems, MES systems, and DMS systems. In addition, this system can be connected to a real controller through an external interface, and can flash the real controller through functional adaptation.

3. Bench Hardware Architecture Design

The hardware architecture design of the automotive software upgrade simulation test bench is the foundation for ensuring testing accuracy and reliability. A reasonable hardware architecture can provide a stable testing environment, support multiple testing scenarios, and have flexible scalability. The following is a detailed analysis of the hardware architecture design of the test bench.

3.1. Processor and Computing Unit

The processor and computing unit are the core components of the hardware architecture of the test bench, responsible for running simulation software and testing scripts, processing test data, and performing real-time calculations. When choosing processors and computing units, factors such as computing power, memory size, and number of I/O interfaces need to be considered. To meet the needs of different testing scenarios, high-performance server level processors or multi-core processors can be used to improve computing efficiency and processing power.

3.2. Interface and Communication Module

The interface and communication module are key components in the hardware architecture of the test bench for connecting with external devices and transmitting data. These modules include CAN bus interface, LIN bus interface, FlexRay bus interface, etc., used for communication with automotive ECU (Electronic Control Unit). In addition, Ethernet interfaces, USB interfaces, etc. need to be provided for data transmission and synchronization with PCs or other testing equipment. To ensure the stability and reliability of communication, it is necessary to use high-quality interface chips and communication protocols, and conduct rigorous testing and verification.

3.3. Sensor and actuator simulation

Sensors and actuators are important components of automotive systems, responsible for collecting vehicle status and executing control instructions. On the simulation test bench, it is necessary to simulate the behavior of these sensors and actuators to simulate the real automotive operating environment. This can be achieved through the use of technologies such as analog circuits, digital signal processors (DSPs), or programmable logic devices (PLDs). In order to simulate the characteristics of different sensors and actuators, corresponding analog circuits and algorithms need to be designed, calibrated and verified.

3.4. Power Management Module

The power management module is responsible for providing stable power supply to the rack hardware and monitoring the power status. In the simulation testing of automotive software upgrades, it is necessary to simulate test scenarios under different voltage and current conditions to verify the compatibility and stability of software upgrades for power management. Therefore, the power management module needs to have functions such as adjustable voltage and current output, overcurrent protection, and short-circuit protection. In addition, it is necessary to provide a power monitoring interface for real-time monitoring of power status and troubleshooting.

3.5. Expandability and modularity

In order to meet the needs of different automotive software and hardware platforms, the hardware architecture of the test bench needs to have flexible scalability and modular design. This can be achieved through the use of modular hardware components, standardized interfaces, and communication protocols. By adding or reducing hardware modules, the testing capability of the test bench can be easily expanded or reduced. In addition, modular design can improve hardware maintainability and reusability, and reduce testing costs.

4. Bench Software and Functional Design

The software architecture design of the automotive software upgrade simulation test bench is the key to ensuring the automation, efficiency, and accuracy of the testing process. A reasonable software architecture can provide an easy-to-use testing interface, support multiple testing scripts and test cases, and have flexible configuration and scalability. The following is a detailed analysis of the software architecture and functional design of the test bench.

4.1. Test Management Software

Test management software is the core part of the test bench software architecture, responsible for managing and scheduling test tasks, monitoring the testing process, and collecting and analyzing test data. The software needs to have a user-friendly interface design that supports the import and execution of multiple test scripts and test cases. In addition, it is necessary to provide real-time testing monitoring functionality to promptly identify and address issues during testing. In order to achieve automation and efficiency in testing, test management software also needs to support automated generation and execution of test scripts, as well as automated analysis and report generation of test results.

4.2. Common functions of OTA on the vehicle side

OTA technology enables vehicles to seamlessly connect with cloud servers through mobile communication interfaces such as 4G, 5G, etc. This connection allows vehicles to receive and install software updates from the cloud in real-time, without the need for users to personally drive to the 4S store or upgrade through other physical means. The application of OTA technology has greatly improved the update efficiency and user experience of automotive software. In the design and development process, it is required that the vehicle must have OTA download function, OTA judgment condition processing function, power-off continuous upgrade function, and software rollback function. This article will provide a detailed analysis of the power-off continuous upgrade function.

During the OTA upgrade process of vehicles, various unexpected situations may occur, such as power outages, insufficient memory, or restarts caused by other system abnormalities. In order to ensure the continuity and stability of the upgrade process, OTA systems have introduced a "breakpoint upgrade" mechanism. When an OTA upgrade is in progress, the system will record the progress and status of the current upgrade task, including downloaded and installed files as well as unprocessed parts. If an unexpected power outage or system restart occurs at this time, when the vehicle is powered on again, the OTA system will first check the status of the current upgrade task. The system will rescan all files included in the upgrade task, and by comparing the installed files with the files to be installed, the system can accurately locate the previously interrupted upgrade section. This step is crucial as it ensures that the upgrade process can continue from the correct position rather than starting from scratch. Once the interrupted segment is identified, the OTA system will continue the upgrade process from that point on. This means that the successfully installed parts will not be reprocessed, saving time and system resources. At the same time, the system will also perform verification to ensure consistency and integrity between the upgrade that continues from the breakpoint and the overall upgrade task. The breakpoint upgrade mechanism not only improves the efficiency of OTA upgrades, but also enhances the reliability and stability of the upgrade process. It ensures that upgrade tasks can be completed smoothly even in the event of unexpected situations, providing users with a better upgrade experience. This mechanism is an indispensable part of modern automotive OTA systems, providing strong support for the continuous updating and optimization of vehicle software. The implementation logic diagram is shown in Figure 1:

4.3. ECU flashing

Unified Diagnostic Services (UDS) is a communication protocol for automotive electronics, which is a network transmission protocol used for device diagnosis in electronic controllers (ECUs). The corresponding standard is ISO 14229-1. This standard is derived from ISO 14230-3 (KWP2000) and ISO 15765-3 (Diagnostic Communication on Controller Area Networks), DoCAN. The term 'unified' in the 'Unified Diagnostic Service' refers to this standard being an international standard, not a specific company's proprietary standard. All newly produced ECUs from first tier suppliers currently support this communication protocol. The electronic controllers in modern cars control a wide range of functions, including fuel injection systems (EFI), engine controllers, transmissions, anti lock braking systems (ABS), door locks, brakes, window actions, and more. When the upper computer or diagnostic instrument requests the unified diagnostic service to the controller, the controller must reply (either positively or negatively) to confirm the fault records in individual control units, update the firmware of the control unit (firmware flashing), interact with the hardware at a lower level (such as turning on or off specific outputs), or perform specific ECU functions. The diagnostic service function is shown in Figure 2:

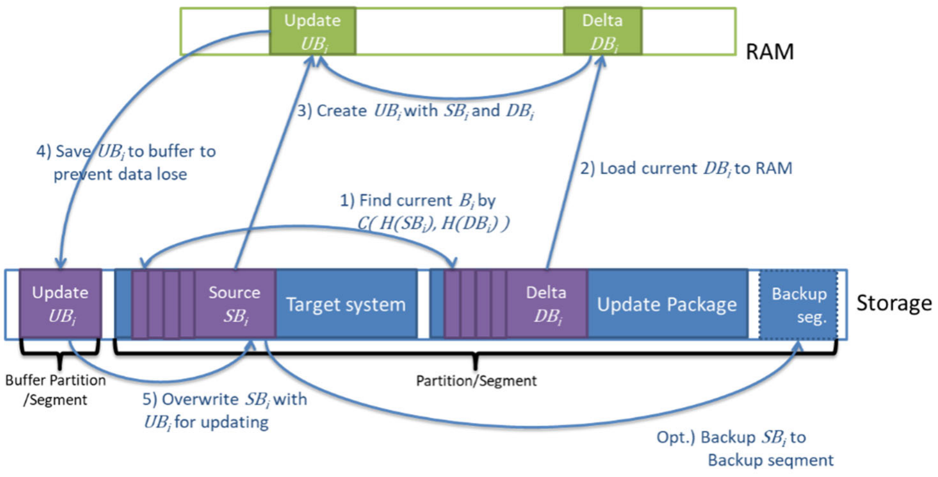
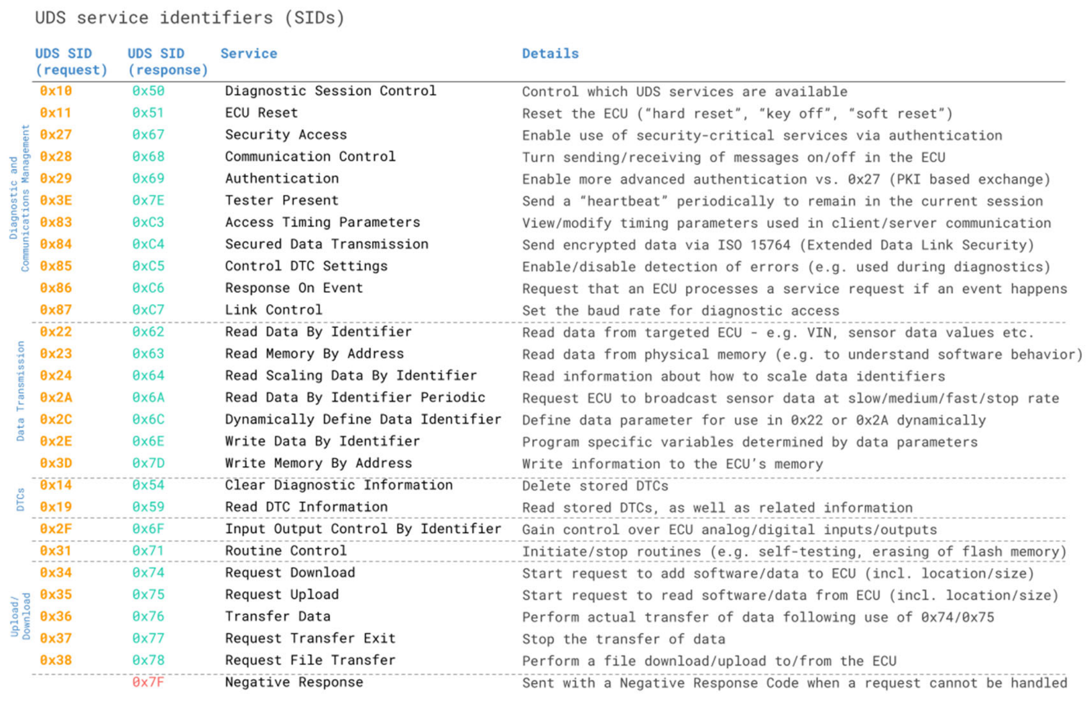

Update UB_i
Delta DB_i
RAM
3) Create UB_i with SB_i and DB_i
4) Save UB_i to buffer to prevent data lose
2) Load current DB_i to RAM
1) Find current B_i by $C(H(SB_i), H(DB_i))$
Update UB_i
Source SB_i
Target system
Delta DB_i
Update Package
Backup seg.
Storage
Buffer Partition /Segment
Partition/Segment
5) Overwrite SB_i with UB_i for updating
Opt.) Backup SB_i to Backup seqment

Figure 1. Power outage and continuous rise mechanism



UDS service identifiers (SIDs)

	UDS SID (request)	UDS SID (response)	Service	Details
Diagnostic and Communications Management	0x10	0x50	Diagnostic Session Control	Control which UDS services are available
	0x11	0x51	ECU Reset	Reset the ECU ("hard reset", "key off", "soft reset")
	0x27	0x67	Security Access	Enable use of security-critical services via authentication
	0x28	0x68	Communication Control	Turn sending/receiving of messages on/off in the ECU
	0x29	0x69	Authentication	Enable more advanced authentication vs. 0x27 (PKI based exchange)
	0x3E	0x7E	Tester Present	Send a "heartbeat" periodically to remain in the current session
	0x83	0xC3	Access Timing Parameters	View/modify timing parameters used in client/server communication
	0x84	0xC4	Secured Data Transmission	Send encrypted data via ISO 15764 (Extended Data Link Security)
	0x85	0xC5	Control DTC Settings	Enable/disable detection of errors (e.g. used during diagnostics)
	0x86	0xC6	Response On Event	Request that an ECU processes a service request if an event happens
	0x87	0xC7	Link Control	Set the baud rate for diagnostic access
Data Transmission	0x22	0x62	Read Data By Identifier	Read data from targeted ECU - e.g. VIN, sensor data values etc.
	0x23	0x63	Read Memory By Address	Read data from physical memory (e.g. to understand software behavior)
	0x24	0x64	Read Scaling Data By Identifier	Read information about how to scale data identifiers
	0x2A	0x6A	Read Data By Identifier Periodic	Request ECU to broadcast sensor data at slow/medium/fast/stop rate
	0x2C	0x6C	Dynamically Define Data Identifier	Define data parameter for use in 0x22 or 0x2A dynamically
	0x2E	0x6E	Write Data By Identifier	Program specific variables determined by data parameters
	0x3D	0x7D	Write Memory By Address	Write information to the ECU's memory
DTCs	0x14	0x54	Clear Diagnostic Information	Delete stored DTCs
	0x19	0x59	Read DTC Information	Read stored DTCs, as well as related information
	0x2F	0x6F	Input Output Control By Identifier	Gain control over ECU analog/digital inputs/outputs
	0x31	0x71	Routine Control	Initiate/stop routines (e.g. self-testing, erasing of flash memory)
Upload/ Download	0x34	0x74	Request Download	Start request to add software/data to ECU (incl. location/size)
	0x35	0x75	Request Upload	Start request to read software/data from ECU (incl. location/size)
	0x36	0x76	Transfer Data	Perform actual transfer of data following use of 0x74/0x75
	0x37	0x77	Request Transfer Exit	Stop the transfer of data
	0x38	0x78	Request File Transfer	Perform a file download/upload to/from the ECU
		0x7F	Negative Response	Sent with a Negative Response Code when a request cannot be handled

Figure 2. Diagnostic Service Function

ECU flashing and firmware upload are carried out in accordance with the specifications of the unified diagnostic service, using the above service functions to flash the ECU. ISO15765-3 is a specification widely used by car manufacturers for UDS services on CAN-BUS, abbreviated as UDSonCAN. The OSI network model defines seven layers of a network, with UDS and UDSonCAN corresponding to the fifth to seventh layers. As shown in Figure 3:

ISSN: 2710-0588

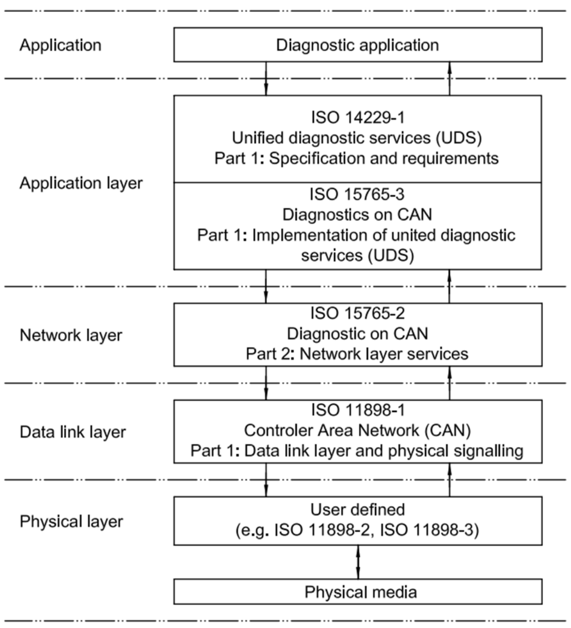

Application
Diagnostic application
Application layer
ISO 14229-1
Unified diagnostic services (UDS)
Part 1: Specification and requirements
ISO 15765-3
Diagnostics on CAN
Part 1: Implementation of united diagnostic services (UDS)
Network layer
ISO 15765-2
Diagnostic on CAN
Part 2: Network layer services
Data link layer
ISO 11898-1
Controler Area Network (CAN)
Part 1: Data link layer and physical signalling
Physical layer
User defined
(e.g. ISO 11898-2, ISO 11898-3)
Physical media

Figure 3. UDSonCAN Service

ISO 15765-2, also known as ISO-TP (TP stands for Transport Layer, Transport Layer), It is an international standard for transmitting network packets over Controller Area Network (CAN). Traditional CAN messages have a maximum of eight bytes of data per frame, while ISO-TP can transmit data exceeding eight bytes. ISO-TP breaks down long data packets into many data frames, and adds some encoded data to facilitate the interpretation of each data frame by the transmitting and receiving ends, and allows the receiving end to restore many data frames to the original message. The maximum payload of a data packet in ISO-TP is 4095 bytes. During ISO-TP operation, one or more bytes from an eight byte CAN data frame are used to store metadata, resulting in a maximum payload of only seven bytes. Metadata is called Protocol Control Information, abbreviated as PCI. PCI may be 1-3 bytes. The first fourth digit indicates the type of frame and indirectly indicates the length of PCI. Figure 4 shows four packet formats used for transmission:

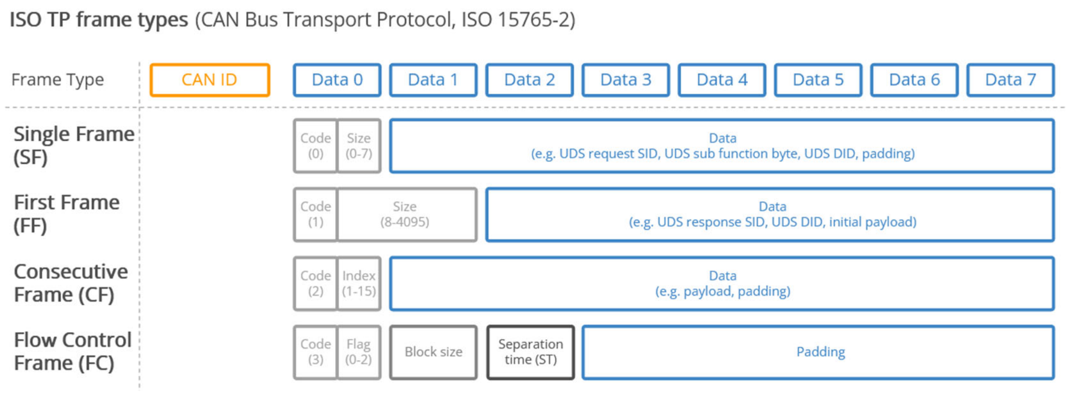


Figure 4. ISO-TP transmission packet format

5. Testing Process and Actual Analysis

The design of the automotive software upgrade simulation test bench ultimately needs to be implemented in the specific testing process and case analysis. The following will introduce a typical testing process and demonstrate the application effect of the test bench through a specific case analysis.

The testing process typically includes the following steps: Test preparation: Prepare test scripts, test cases, and test environments based on testing requirements. Ensure that all necessary hardware and software are installed and configured correctly.

Test execution: Run test scripts to simulate a real car operating environment and conduct comprehensive testing of the software upgrade process. Monitor the testing process, record test data and logs.

Test result analysis: Analyze and process the collected test data, evaluate the effectiveness of software upgrades, and identify problems. Generate test reports, summarize test results and recommendations.

Problem fixing and validation: Fix the identified issues and retest for validation. Ensure that all issues are properly resolved and the quality and reliability of software upgrades are improved.

Our company is upgrading the software of the in car entertainment system of a certain domestic car model on the designed platform to improve the stability and user experience of the system. In order to ensure the safety and reliability of the upgrade, it has been decided to use a simulation test bench for testing.

Test preparation: Test scripts and test cases have been prepared according to the testing requirements, including functional verification, performance testing, compatibility testing, etc. At the same time, corresponding hardware and software environments have been configured to ensure that all necessary components have been correctly installed and configured.

Test execution: Run test scripts to simulate a real car operating environment and conduct comprehensive testing of the software upgrade process of the in car entertainment system. Monitor the testing process, record test data and logs. Several issues were discovered during the testing process, such as slow system startup speed and unstable audio output.

Test result analysis: Analyze and process the collected test data, evaluate the effectiveness of software upgrades, and identify problems. Generate a test report that summarizes the test results and recommendations. Detailed analysis and localization were conducted on the identified issues, and the root cause of the problem was identified.

Problem fixing and validation: The identified issues were fixed and retested for validation. After multiple iterations of testing, all issues were ultimately resolved, ensuring an improvement in the quality and reliability of the software upgrade. Through simulation testing on the test bench, the car manufacturer has successfully completed the software upgrade of the in car entertainment system and applied it to actual vehicles.

6. Conclusion

This article discusses in detail the design of a simulation test bench for automotive software upgrades and proposes an integrated testing platform solution. This platform can simulate real car operating environments and conduct comprehensive testing of software upgrade processes, including functional verification, performance testing, compatibility testing, etc. The effectiveness and practicality of the test bench have been demonstrated through practical application cases.

References

[1] Exploration into the Construction of Automotive Software Upgrade System Yang Lingyun; Chang Yongbin; Zhang Ni; Su Junfang China Automotive, 2023 (03)

[2] DiBa Computer Anti Virus Service Network Special Software Upgrade Information Table Computer Programming Techniques and Maintenance, 1994 (05)

[3] Software Upgrade Introduction Desktop Publishing and Design, 1995 (05)

[4] Application of Certified Software Upgrade Scheme in Small Embedded Systems Wang Tongzhu Information and Computer (Theoretical Edition), 2022 (14)

[5] Multi objective Optimization Method for Software Upgrade Problems Zhao Songhui; Ren Zhilei; Jiang He Computer Science, 2020 (06)

[6] Design of an Embedded Communication Power Controller Software Upgrade System Jiang Xiaodong Integrated Circuits and Embedded Systems, 2024 (10)

[7] Design and Implementation of Software Upgrades Based on Distributed Data Communication Devices Zhang Fengzhe; Chen Peng Optical Communication Research, 2018 (01)

[8] Application Software Version Upgrade Testing Research Jiang Wen; Liu Likang Microcomputer Application, 2021 (07)

[9] A wireless software upgrade method for mine positioning equipment Zhang Lifeng; Jin Yeyong; Chen Kang; Wang Wei; Zhou Shu Industrial and mining automation, 2019 (11)

[10] upgraded "Cat" with software Tian Qiu Microcomputer World, 1999 (05)