Design Method of On-orbit Software Injection Based on the Microprocessor BM3803 with High Reliability

Zhiyan Lyu[1(✉)], Jinfang Dou[1], Pinyao Song[2], and Haolin Zhou[2]

[1] Xi'an Institute of Space Radio Technology, Xi'an 710100, China
lvzy71@sina.com
[2] The University of Melbourne, Carlton, Melbourne, VIC 3053, Australia

Abstract. On-orbit software injection function is a necessary means for on-orbit software maintenance of increasingly intelligent space-borne equipment. Based on the characteristics of the microprocessor BM3803, this paper presents a high reliability on-orbit software injection design method with strong error detection and correction, and self-locking prevention ability. Each bit of all of program data can be detected and corrected by bit-by-bit-two-out-of-three when it is wrong. The Bootstrap Programs achieve the purpose of loading conversion between memories with different bit widths, and EDAC. If the upgraded program cannot run, with the assistance of the FPGA the bootstrap programs can make the application return to the original program, so as to restore the bus communication link and basic functions. Experiments shows that this Design method can improve the reliability of system and make function expanding possible, also proves the feasibility of the scheme. The scheme has a good guiding significance for the on-orbit software maintenance design of space-borne equipment based on the microprocessor BM3803.

Keywords: BM3803 · On-orbit software injection · Bit width conversion · Self-locking prevention · Error detection and correction · High reliability

1 Introduction

With the increasing requirement of satellite users and the rapid development of space electronic technology, the shorter launch cycle, the more complex functions of space-borne software, the shorter time of design and development and testing, the worse space electromagnetic environment, and the more urgent requirement is that space-borne equipments have an on-orbit software maintenance [6] capability. Most of the satellites developed by NASA and ESA abroad have an on-orbit injection function [6, 7]. In recent years, many subsystems [2] of domestic satellites have an on-orbit software injection function [1–4, 9], such as Beidou satellites [3], many formation flying satellites and so on. Practice has proved that on-orbit software injection function is an effective means to solve problems in on-orbit flight.

At present, the degree of localization of domestic aerospace processors is getting higher and higher, and BM3803 is more and more widely used in aerocrafts. It is necessary to operate and maintain aerospace products based on microprocessor

Y. Wang et al. (Eds.): *Signal and Information Processing, Networking and Computers*, LNEE 628, pp. 461–469, 2020.
https://doi.org/10.1007/978-981-15-4163-6_55

BM3803 with high reliability on orbit. Reference [2] describes an on-board injection method based on vector table for microprocessor BM3803. This method can correct the problems of existing systems, but it lacks consideration of its own security and reliability. Considering the harsh space electromagnetic operation environment, this paper proposes a design method of on-orbit injection maintenance with better self-locking prevention solutions and anti-single particle overturning error detection and correction measures, which has good guiding significance for the on-orbit injection design of satellite-borne equipment based on the microprocessor BM3803.

2 Design Ideas

2.1 Storage Device Selection

BM3803 is a radiation hardening domestic 32-bit space processor based on SPARC V8 architecture. It does not have its own memory. The satellite-borne equipment based on the microprocessor BM3803 must expand its program memory and random-access memory. In order to cope with the relatively harsh electromagnetic environment in space, the hardware design of satellite-borne equipment should select devices with strong radiation resistance and single-particle immunity as much as possible when selecting storage devices. PROM, as an aerospace storage device, has single-particle immunity. However, because it has small storage capacity, it cannot be programmed online, and its use conditions are limited, so PROM with single-particle immunity is selected as a storage device with no on-orbit maintenance requirement, single function, clear task, and small program scale. At present, there is no space-class memory device with strong radiation resistance, single-particle immunity and large capacity. As for application software, due to its powerful function, the demand may change and the requirement for storage capacity is large. As the on-orbit software injection function requires storage with on-line programming capability, NorFlash devices with on-line programming and large storage capacity are generally selected.

At present, aerospace PROM(HS1-6664RH-Q) is an 8-bit wide memory. NorFlash has 8-bit wide and 16-bit wide memories. Since BM3803 is a 32-bit processor, NorFlash selects 16-bit wide devices, SRAM selects 32-bit wide or splices 8-bit or 16-bit SRAM into 32-bit SRAM for use in order to improve operation efficiency.

2.2 Selection of On-orbit Injection Mode

The on-orbit injection of the program ways include the whole software injection [5, 7] and the function module injection [1–4]. In multitasking applications supported by multitasking operating systems, the data flow and control flow between software components are mainly realized through message queues and binary semaphores. The connection between software components is relatively “loose”. The execution of each components of a program is driven by events, and there is no particular order or necessary connection between the components of a program. This case is suitable for injecting function modules to correct fault modules, correct components do not need to be changed, and the amount of data to be injected is greatly reduced. However, for

control software without operating system support, the data flow and control flow between components are mainly transmitted through global variables and parameters. The components in the software are closely linked, and the execution between program components is carried out in sequence. Generally, the on-orbit injection of the program adopts the whole program injection. This paper is an introduction to the high reliability design method for the whole program on-orbit injection.

2.3 On-orbit Injection Process and Precautions

The on-orbit injection process of satellite-borne software: via the sky-earth data transmission channel, the injection program data is transmitted from the ground station to the satellite via the uplink channel, and the injection program data is transmitted to satellite data management equipment after receiving. The data management equipment transmits the injection program to the corresponding equipment via the satellite internal network according to the agreed communication protocol. After the program (also called an original program, burned into program memory of the satellite-borne equipment when on the ground) loaded into a random access memory of the equipment receives the injection data packets of the other program which improved, checks the data of the injection program (including checking the correctness of the frame format, bit error, error code and so on). After the check is correct, burns the injection program data online into the non-volatile online programmable memory (Flash or EEPROM). The storage area is agreed in advance. Then power off and re-power on the device. The boot program determines whether to boot the original program or the injection program according to the corresponding boot strategy. After the boot is completed, jump the program pointer to the starting position of the program that is loaded into SRAM.

The key to the successful design of satellite-borne software on-orbit injection is the design of this boot program and the design of receiving, verifying and on-line burning the data of the injection program. Therefore, the following matters should be paid attention to in these design links:

Firstly, in order to adapt to the higher communication rate, before the receiving of the upper injection program is completed, no check is carried out, only the upper injection data of each frame is received into the memory area, and after all the upper injection frames are received, the check of each frame and the check of the whole upper injection program are carried out.

Secondly, the frame format check and data check shall be carried out according to the format and check method agreed in advance, and the check result shall be telemetered back to the earth in time, so that flight control personnel on the earth can know the errors in the injection process in time and correct them.

And finally, considering the relatively harsh electromagnetic environment in space and the radiation-resistant ability of NorFlash, three copies of programs are burned when stored in NorFlash, and three take two comparisons when loaded, so as to ensure that the correct programs can be loaded after single event multi-bit upsets. The program should have error detection and correction capability with regularly.

2.4 Boot Software Design

Based on space environment and security considerations, the boot program is stored in PROM(HS1-6664RH-Q), the application software is stored in 16-bit wide NorFlash, and the final application software needs to be loaded into 32-bit SRAM to run. Both 8-bit PROM(HS1-6664RH-Q) and 16-bit NorFlash are located in the PROM address area of BM3803, while the initial bit width value of the PROM address area of BM3803 is configured by the 1st and 0th bits of GPIO, which will be automatically set to 8-bit width by FPGA at power-up. Such a boot program stored in the PROM address area and running here cannot read the correct 16-bit data from the 16-bit wide NorFlash in the PROM address area that has been configured to be 8-bit wide. Therefore, it is impossible to design a boot program for this correct boot, which requires the design of two boot programs, namely, bootloader 1 and bootloader 2. The function of bootloader 1 is to load bootloader 2 into a 32-bit wide SRAM. The function of bootloader 2 is to load the application software into 32-bit wide SRAM. Both bootloader 1 and bootloader 2 are stored in 8-bit PROM(HS1-6664RH-Q), but their running environments are different. Bootloader 1 runs in 8-bit PROM(HS1-6664RH-Q), while bootloader 2 runs in 32-bit wide SRAM. The bootloader 1 loads the bootloader 2 from the 8-bit PROM(HS1-6664RH-Q) into the 32-bit SRAM, and then the program pointer jumps to the starting position of the bootloader 2 loaded into the 32-bit SRAM to start running the bootloader 2. The bootloader 2 can reset the bit width of the PROM address area of the BM3803 to 16 bits, thus correctly reads out and loads the application software from the 16-bit NorFlash into the 32-bit SRAM, and after loading is completed, the program pointer jumps to the starting position of the application software loaded into the 32-bit SRAM to run the application software. So as to achieve the purpose of loading conversion between memories with different bit widths.

2.5 Reliability Design

Considering the relatively harsh electromagnetic environment in space and the radiation resistance of NorFlash, the correct program can be loaded in case of single event upset, and the program can be provided with regular error detection and correction capability.

In order to ensure that the program can be loaded correctly even if the program data are knocked over by single particles, it is necessary to burn three copies in NorFlash during online programming of the injection program. The bootloader 2 has the function of reading the application software in NorFlash based on bit-by-bit-two-out-of-three. The specific strategy is: when the bootloader 2 reads the application software from the 16-bit NorFlash, the bootloaer 2 reads the data from the loaded starting addresses of the three regions respectively, and makes a judgment of “2-out-of-3”. If at least two copies are the same, one copy of two copies is taken; if the three copies are different, each bit of the 16-bit data read from NORFlash is judged by “2-out-of-3”, then is combined into new 16-bit data. Finally, the combined data is taken as the final result, the combined data is stored in the SRAM backup area, the entire program is read out according to the loaded length in sequence, and finally, the combined program stored in the SRAM backup area is loaded into the SRAM running area. First of all, according to the 16-bit

whole, the ratio loading efficiency is greatly improved, and the program safety is improved to the greatest extent in case of failure by bit-by-bit-two-out-of-three.

2.6 Anti-self-locking Design

Boot strategy is also a very important factor for the success of the on-orbit injection function. Incomplete boot strategy may result in the loss of function of the whole equipment after on-orbit injection, thus causing the failure of the whole subsystem and even the scrapping of the whole satellite. In order to avoid this situation, the on-orbit injection boot strategy must have anti-self-locking measures. Anti-self-locking boot method is similar to windows-startup with manual intervention and automatic selection. This method needs ground remote control, satellite-borne data management equipment, FPGA software of the corresponding satellite-borne equipment and the boot program of the corresponding satellite-borne equipment cooperate. After the satellite-borne equipment is powered on, there is no normal telemetry return within a period of time. The ground flight control personnel can conclude that the injection program fails to operate normally and the communication link cannot be established. It is necessary to switch to the original program. The ground flight control personnel sends the switch to the original program instruction to the data management equipment. After receiving the instruction, the data management equipment sends the satellite-borne equipment 4 times of satellite-borne equipment reset instruction (pulse instruction, no more than 8 times) within 1 s. After receiving four reset instructions, the FPGA of the satellite-borne device concludes that the ground flight control personnel intends to switch to the original program, then gives instruction to use the original program through the IO interface while resetting the central processor BM3803 of the satellite-borne equipment (0x55555555 indicates that the original program is used, and other values indicate that the loaded program version is determined by the bootloader 2 of independent strategy). After resetting, BM3803 runs the bootloader 1 from PROM address area to load the bootloader 2 into SRAM. Then the program pointer jumps to the entry position of the bootloader 2 in the SRAM to run the bootloader 2. The bootloader 2 first reads the instruction mark given by FPGA through the I/O interface, and judges whether to use the original program according to the instruction mark. i.e. whether the instruction mark is 0x55555555, if so, the bootloader 2 loads the error-corrected original program in bit-by-bit-two-out-of-three error detection and correction mode, otherwise the bootloader 2 adopts an autonomous boot strategy. First, the bootloader 2 reads the data of the injection-mark position to judge whether there is an injection program (i.e., the data of the injection-mark position is 0x000055AA). If there is, the bootloader 2 loads the error-corrected injection program in bit-by-bit-two-out-of-three error detection and correction mode, otherwise, loads the error-corrected original program in bit-by-bit-two-out-of-three error detection and correction mode. This method solves the problem that the error can not be corrected or even the equipment function fails after the injection program start up every time, though the verification of injection program data is correct but the injection function is actually incorrect by a strange combination of circumstances.

3 Verification

Based on a satellite remote controller product development and test platform of a remote sensing satellite, the high-reliability on-track injection method of the program was tested in a laboratory environment. The test connection block diagram is shown in Fig. 1. The onboard device program memory and program running space allocation map are shown in Fig. 2.

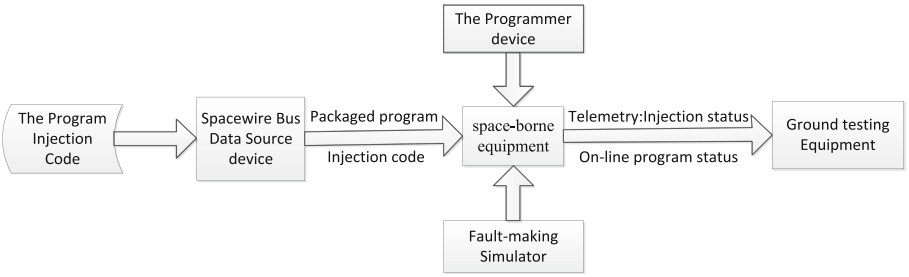

Fig. 1. Test connection block diagram

Program Storage Area Allocation

0x00000000	bootloader1	PROM
	bootloader2	PROM
0x00002000	reserve	FLASH
0x10100000	Injection program storage area	FLASH
0x10250000	Original program storage area	FLASH

Program Running Area Allocation

0x00000000	Bootloader1 running area	PROM
0x40000000	Bootloader2 running area	SRAM
0x40004000	On-board software running area	SRAM

Fig. 2. The storage map of space allocation

The Spacewire bus data source device splits the program injection code into multiple data blocks. After framing each data block according to the agreed protocol format, the data packets are transmitted to the onboard device through the spacewire bus.

The fault-making simulator makes mistakes in the original program or the injected program stored in the on-board device for error detection and correction verification.

The ground testing equipment simulates the onboard data management device to send commands and receive telemetry information to confirm the feasibility of the on-track injection scheme through the 1553B bus.

3.1 On-orbit Injection Scheme Verification

The program is transmitted to the onboard device through the spacewire bus data source device. Through telemetry, it can be known that the injected data packets are successfully received and verified, online programming is successful, the bootloading is successful, and the injected program runs normally (see Fig. 3). A total of more than 30 on-orbit injection tests were performed, and each was successful.

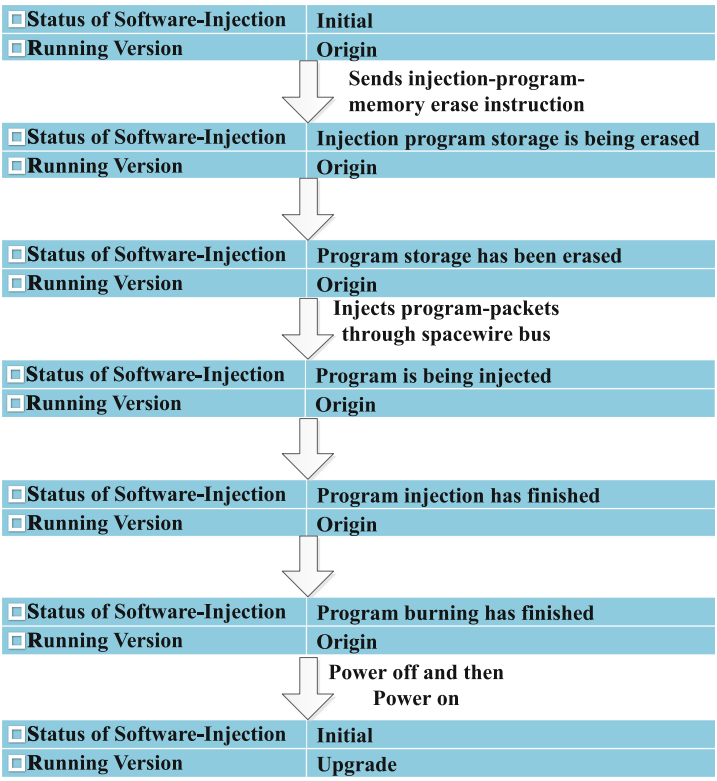

Fig. 3. On-board software on-orbit injection process telemetry screenshot

3.2 Anti-self-locking Scheme Verification

The ground testing equipment simulates the satellite data management device to send a reset signal to the onboard controller equipment for 4 times within one second. When the onboard equipment restarts, the telemetry display current running program version is the original program (see Fig. 4). After 100 tests, each time you can successfully switch from the injected program to the original program to run.

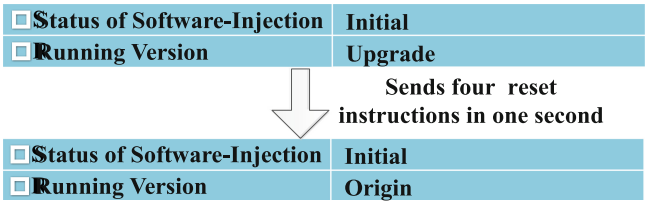

Fig. 4. Anti-self-locking scheme verification telemetry

3.3 Error Detection and Correction Verification

In order to verify the error detection and correction function of bootloader softwares, The fault-making simulator is used to make mistakes to the injected program stored in the onboard equipment. Then the onboard equipment is restarted by powering up. According to the telemetry we can know that the injected program runs normally, and by the DSU interface we can know that the wrong data in the memory has been corrected (see Fig. 5).

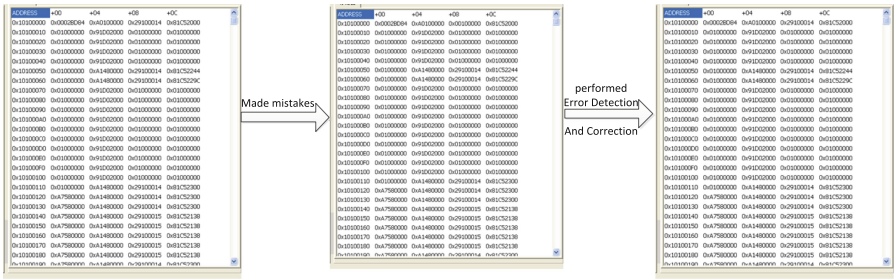

Fig. 5. Program data in memory

The design has been verified in the laboratory environment and during the satellite electrical performance test, and has been applied to on-board equipment of several types of satellites.

The above test shows that the design method of on-orbit injection has high reliability, strong error detection and correction capability, the implementation method is ingenious and the scheme is feasible.

4 Summary

Having the function of on-orbit injection of programs is a trend in the future for all equipment of aerocraft with higher and higher intelligence. In addition to effectively solving problems in flight, it can even improve or change the function of equipment, which is worth our time to study. In addition, While evolvability itself can be viewed as on-board perfective maintenance, it necessitates preventive maintenance and corrective

maintenance for eliminating or mitigating potential error conditions caused by residual faults in an upgraded system configuration or software version [7], so the form of on-orbit injection, safety and reliability of on-orbit injection also need further research and discussion.

References

1. Li, Z., Du, J., Dang, J.: Implementation of on-board software maintenance for spacecraft based on DSP. Aerosp. Control Appl. **43**(6), (2017)
2. Zhu, Y., Wang, J., Shi, Z., Yang, M.: Research the method of on-board reprogramming based on vector table. Electron. Des. Eng. (2013)
3. Wu, G., Zhong, X., Tao, X.: Interrupt processing technology of on-board software reconfiguration for satellites. Space Electron. Technol. (2), 68–71 (2011)
4. Wang, H., Wang, H., Jin, Z.: Rollback-able on-board software upgrade method based on incremental link. J. Zhejiang Univ. (Eng. Sci.) (4), (2015)
5. Li, Y., Ji, F., Huang, Y.: The design of on-board programming for satellite DSP software. Guid. Fuze **32**(4), (2011)
6. Peccia, N., Giannini, F.: XMM instrument on-board software maintenance concept. NASA report (1), 984–992 (1994)
7. Thi, A.T., Tso, K.S., Alkalai, L., Chau, S.N., Sanders, W.H.: On-board guarded software upgrading for space missions. In: Proceedings of the 18th Digital Avionics Systems Conference, St. Louis, MO, 24–29 October 1999 (1999)
8. Tai, A.T., Tso, K.S., Alkalai, L., Chau, S.N., Sanders, W.H.: Low-cost error containment and recovery for on-board guarded software upgrading and beyond. IEEE Trans. Comput. **51**(2), 121–136 (2002)
9. Wei, Y., Dong, Z., Zhang, H.: Method of embedded onboard software upgrading based on file. Microcontroll. Embed. Syst. (5), (2018)