一种基于微小卫星系统软件在轨编程功能的设计方法

钱方亮[1,2] 林荣锋[2] 周 宇[2] 张增安[2] 张 磊[2]

[1](上海交通大学电子信息与电气工程学院 上海 201109)

[2](上海航天控制技术研究所 上海 201109)

摘 要 高集成、短周期、低成本的研制要求使微小卫星系统软件的在轨编程方案难以继承和沿用于传统大型卫星。设计一种适合于微小卫星系统软件,针对 SRAM + FLASH 存储器配置的可编程方案,结合对数据包设计、存储分配以及编程功能机制的充分思考,实现在轨编程功能要求。设计满足卫星系统软件在轨修复、功能变更以及扩充的需求,从而提高卫星系统的可靠性和安全性,增强系统功能的灵活性。该设计方案比现有方案具有硬件配置要求低、与原软件完全隔离、操作便捷、容错能力高、移植性好等特点。

关键词 在轨编程 系统软件 微小卫星

中图分类号 TP311 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2018.12.004

DESIGN OF ON-ORBIT PROGRAMMING FOR MICROSATELLITE SYSTEM SOFTWARE

Qian Fangliang[1,2] Lin Rongfeng[2] Zhou Yu[2] Zhang Zeng'an[2] Zhang Lei[2]

[1](*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 201109, China*)

[2](*Shanghai Aerospace Control Technology Institute, Shanghai 201109, China*)

Abstract Development requirements such as highly-integrated, short-cycle, and low-cost make it difficult for the on-orbit programming solution of microsatellite system software to inherit and follow traditional large satellites. We designed a programmable scheme for SRAM and FLASH memory configuration, which was suitable for microsatellite system software. Combining with the full consideration of data packet design, storage allocation and programming function mechanism, the requirements of on-orbit programming function were realized to meet the needs of on-orbit repair, functional change and expansion of satellite system software, so as to improve the reliability and security of satellite system and enhance the flexibility of system function. Compared with existing schemes, the scheme has the characteristics of low requirements for hardware configuration, complete isolation from the original software, convenient operation, high fault tolerance and good portability.

Keywords On-orbit programming System software Microsatellite

0 引 言

随着组网式、链路式或星座集群式的卫星协同模式发展浪潮的到来,传统卫星构架和研制模式已经无法适应当前商业市场对卫星产业规模和成本的要求。卫星的小型化、高集成度、多功能性的发展方向势在必行。而系统软件的集成化设计是实现卫星小型化、低成本的重要手段之一。与此同时对微小卫星软件复杂度、灵活性、可靠性等要求也随之大幅提高。

重构容错技术[1]利用可重用的软硬件资源,实现卫星在轨的故障检测、缺陷修复、功能变更和任务扩展,对保证卫星在轨的正常工作、寿命延长、响应各类需求具有重要意义。由于卫星故障的多样性,重构容

收稿日期:2018 - 04 - 22。**钱方亮**,工程师,主研领域:航天器软件设计和开发。**林荣锋**,高工。**周宇**,工程师。**张增安**,高工。**张磊**,工程师。

错技术的研究实现方向也很广泛。如通过数据读写比对 SRAM 区进行检测,并利用 PROM 的引导软件进行重载以实现 SRAM 的分区重构[2];通过工程经验,对可能出现的故障情况进行软件自主检测诊断并通过预设的故障模式实现系统的重构[3];基于 FPGA 实现对存储分区引导重构[4]等。

软件在轨编程功能是用于卫星缺陷修复、功能变更和扩展,实现系统重构容错最重要的手段。早期的星载计算机多用于 SRAM + PROM 的存储构架,系统软件固化于不可擦写的 PROM 中,在轨编程功能设计仅基于对 SRAM 区域[5-7]。但由于 SRAM 空间有限,对上注模块的数量和规模有严格的限制。另外当发生断电或复位,编程模块需要重新上注,难以重复使用。而基于 DSP 平台利用在 PROM 区预置钩子函数实现部分模块替代的编程方案[8],由于钩子数量有限且具有局限性从而仅能实现部分模块的增加,无法做到任意模块的变更。而目前采用 SRAM + PROM + EEPROM(FLASH)存储构架的计算机,利用存放于 PROM 的引导软件实现对 EEPROM 或 FLASH 区软件或补丁的搬运和执行工作[9-11]。在早期方案基础上增加了对 EEPROM(FLASH)区的编程功能。但对存储器配置要求较高,而且增加了引导软件的实现需求,提高了研制成本和时间。因此针对这样的情况,一种适用于小型卫星,实现卫星在轨功能修复、变更、扩充的远程编程技术的研究显得尤为重要。

1　SRAM + FLASH 的软件构架

本文提出一种基于 SRAM + FLASH 存储构架的在轨编程功能设计方案。此方案不仅能够实现卫星系统软件在轨编程功能的要求,而且功能实现的硬件要求较低,无需增加额外的存储器配置,降低了研制成本,同时无需开发引导软件,减少了软件复杂性和研制周期。

1.1　处理器存储构架及软件运行

本文以 TMS570LS31x 系列控制器为例进行详述,该控制器采用 ARM Cortex-R4F 处理器,片内配备 256 KB 的 RAM 以及 3 MB 的 FLASH 存储器,且均具有 ECC 校验功能。

根据该控制器的硬件环境,系统软件固化在 FLASH 中。如图 1 所示,当计算机上电或复位后,程序将从存储器 0x0 地址,即 FLASH 区域开始运行,通过对 FLASH 区代码指令的读取以及 SRAM 区数据的操作完成程序的正常工作。

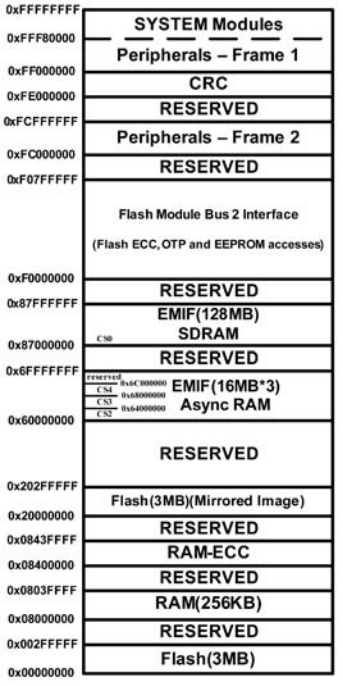

图 1　存储空间分布

1.2　系统软件的存储设计

由于本方案不添加引导功能且不增加片外存储配置,仅采用片内存储空间实现星载软件各类功能,那么,FLASH 内的代码存储分配以及各项功能设计是实现星载软件应用程序、在轨编程维护功能、保证系统可靠性和安全性的关键。

如表 1 所示,片内 FLASH 区分为三个部分,而实际可用空间为 Bank0 和 Bank1 两部分,共由 27 块子区域组成。每一块子区域分别对应一个“写保护锁”,因此通过对空间各区域存储内容的设计,实现 FLASH 区可修改区和不可修改区的划分,确保对程序数据的有效保护,防止误擦误写的操作,提高系统软件的可靠性和安全性。

表 1　FLASH 区 Bank 和 Sector 分布

Sector NO.	Sector Size	Low Address	High Address
Bank 0: 1.5 MB			
0	32 KB	0x0000_0000	0x0000_7FFF
1	32 KB	0x0000_8000	0x0000_FFFF
2	32 KB	0x0001_0000	0x0001_7FFF
3	32 KB	0x0001_8000	0x0001_FFFF
4	128 KB	0x0002_0000	0x0003_FFFF
5	128 KB	0x0004_0000	0x0005_FFFF
6	128 KB	0x0006_0000	0x0007_FFFF
7	128 KB	0x0008_0000	0x0009_FFFF
8	128 KB	0x000A_0000	0x000B_FFFF
9	128 KB	0x000C_0000	0x000D_FFFF

续表 1

Sector NO.	Sector Size	Low Address	High Address
10	128 KB	0x000E_0000	0x000F_FFFF
11	128 KB	0x0010_0000	0x0011_FFFF
12	128 KB	0x0012_0000	0x0013_FFFF
13	128 KB	0x0014_0000	0x0015_FFFF
14	128 KB	0x0016_0000	0x0017_FFFF
Bank 1: 1.5 MB			
0	128 KB	0x0018_0000	0x0019_FFFF
1	128 KB	0x001A_0000	0x001B_FFFF
2	128 KB	0x001C_0000	0x001D_FFFF
3	128 KB	0x001E_0000	0x001F_FFFF
4	128 KB	0x0020_0000	0x0021_FFFF
5	128 KB	0x0022_0000	0x0023_FFFF
6	128 KB	0x0024_0000	0x0025_FFFF
7	128 KB	0x0026_0000	0x0027_FFFF
8	128 KB	0x0028_0000	0x0029_FFFF
9	128 KB	0x002A_0000	0x002B_FFFF
10	128 KB	0x002C_0000	0x002D_FFFF
11	128 KB	0x002E_0000	0x002F_FFFF
Bank 7: 64 KB, dedicated for EEPROM emulation			
0	16 KB	0xF020_0000	0xF020_3FFF
1	16 KB	0xF020_4000	0xF020_7FFF
2	16 KB	0xF020_8000	0xF020_BFFF
3	16 KB	0xF020_C000	0xF020_FFFF

针对微小卫星功能要求以及目前星载软件的规模估计,如图 2 所示对 FLASH 区存储空间进行分配。将片内 3 MB FLASH 划分成 9 部分,分别用于三份应用软件的备份存储、重要数据的保存、以及编程补丁的固化。由于片内 FLASH 是由两块 1.5 MB 的 FLASH 块拼装设计实现的,设计确保相同的三份代码分别存放在两块中,以提高存储空间的可靠性。

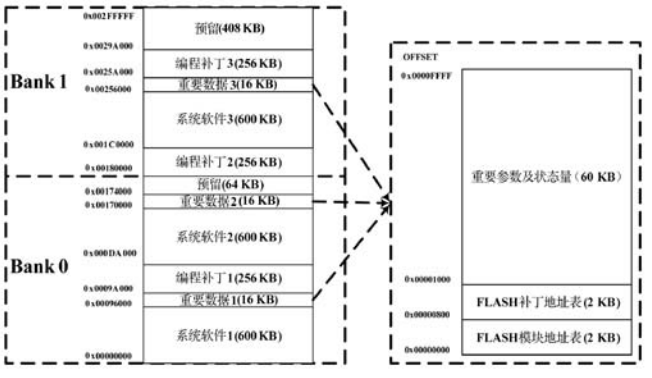

图 2 星载计算机存储空间分配

应用软件区:为避免空间环境对存储器造成瞬间位翻转效应或损坏,对星载软件进行备份存储。若存储器空间发生局部翻转或损坏,可通过 3 份程序的“三取二”比对功能,实现存储数据的及时纠错校正。

重要数据区:对星上重要数据和状态量进行保存。由于掉电或复位后,SRAM 区数据无法保存。因此表征软件运行状态,例如软件首次上电标志、补丁区自动启用开关等重要信息将存储在 FLASH 中备份。除此之外,用于在轨编程使用的模块地址表以及补丁地址表也存放在此区域。

编程补丁区:当确认编程内容工作正确有效后,可将具有长期使用需求的在轨编程包固化在 FLASH 中,确保发生复位后,软件补丁无需重新上注,及时得到使用。

卫星在轨期间,应用软件区与重要数据区两部分区域,除“三取二”维护操作外,不可通过注数修改,以确保原系统软件的可用性。

2 软件模块地址表实现

软件模块地址表用于存储系统软件各模块的首地址。地址表的大小可根据软件的实际规模进行增减。当前地址表大小预设为 2 048 字节,表内每一个单元占用 4 个字节(32 位),因此地址表最多可供 512 个可编程模块存放入口地址。

每个模块入口地址在地址表的位置固定且唯一,如图 3 所示,地址表的首地址即为模块索引 Id 为 1 模块的入口地址,以此类推,地址表末地址即为模块索引 Id 为 512 模块的入口地址。

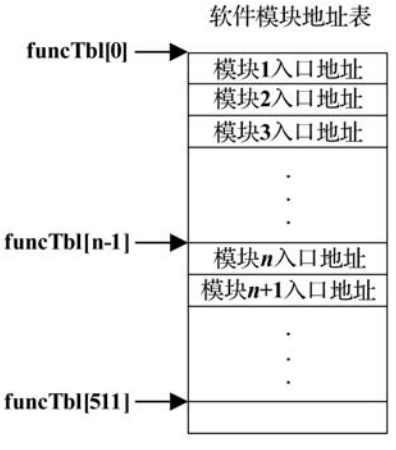

图 3 模块地址表的实现

2.1 模块及补丁地址表设计

为实现系统可编程模块的建立、使用以及维护功能,软件在 FLASH 和 SRAM 指定区域内分别定义和维护一份模块地址表。FLASH 区地址表用于模块地址的记录和地址表的建立。SRAM 地址表用于模块调用

地址查询和地址表更新。此外,根据可编程内容在 FLASH 区维护一份 FLASH 区补丁地址表,用于存放需要替换原模块入口地址的替换信息。

2.2　地址表工作机制

当计算机发生上电或复位时 FLASH 区地址表将被重新建立,建立机制及调用关系如图 4 所示。首先,对 FLASH 区的模块地址表进行初始化,将各可编程模块的入口地址依次写入 FLASH 区地址表中;然后,将 FLASH 区地址表搬到 SRAM 预设的地址表首地址中,并将可编程模块的调用关系与 SRAM 地址表绑定;最后,根据指令查找补丁地址表内是否存在模块替换信息,如果存在,则记录表中替换单元的地址偏移以及地址数据,更新 SRAM 区模块地址表相同偏移模块的入口地址。

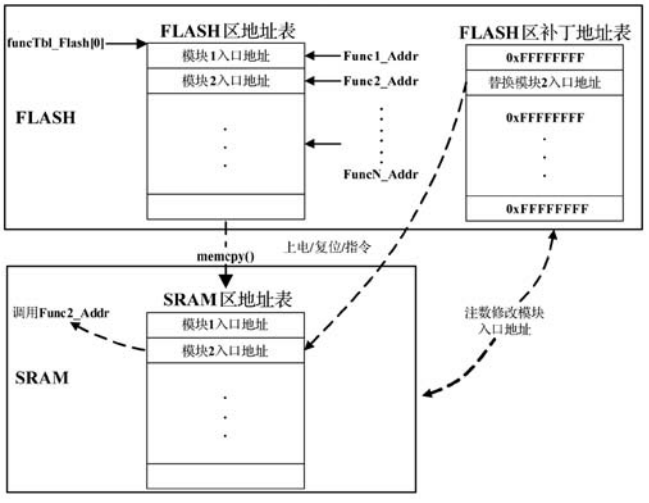

图 4　模块地址表的建立及调用机制

在软件运行中,当某个可编程模块被软件调用时,利用 SRAM 地址表获取该模块的入口地址,通过地址跳转完成模块的调用。

3　在轨编程功能实现

在轨编程功能不仅可以实现软件的故障修复,还可以根据实际情况,完成卫星功能变更和增减,从而提高卫星运行的可靠性和灵活性。该功能分为两部分:SRAM 区在轨编程和 FLASH 区在轨编程。

由于 SRAM 存储具有易失性,因此 SRAM 区可编程功能用于满足卫星功能短期变更或者长期变更前在轨验证需求。当编程内容存在错误或无需继续使用,则可以通过指令上注或复位的手段实现可编程内容的快速清除。与之相对应,利用 FLASH 存储的非易失性,FLASH 区可编程功能可用于存放充分验证后且具有长期变更需求的补丁。当发生复位后,存放于 FLASH 区的补丁通过注数指令即可实现变更功能的再次运行。

3.1　编程数据包设计

通过预先定义的数据包格式以及地面工具完成编程数据包的生成。如图 5 所示,编程数据包由包头、包类型、包序号、数据区以及校验码等信息组成。各部分信息分别用于检测数据包的正确性、识别包的类型、携带包索引信息以及承载编程数据有效信息。

数据内容部分包含数据写入的地址信息、有效数据个数以及有效数据。由于数据包长度固定,当有效数据个数小于数据包可容纳的数据个数时,则根据有效数据个数写入实际数据数量,其余以填充码进行填充。

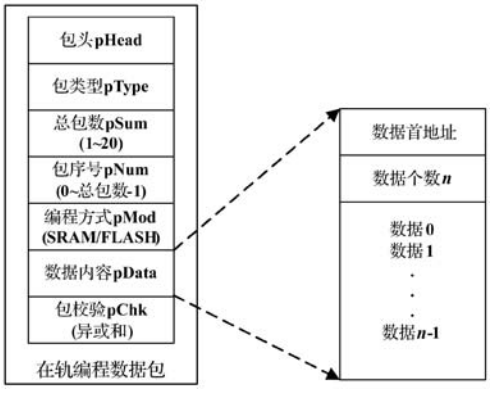

图 5　在轨编程数据包格式

3.2　编程数据注入

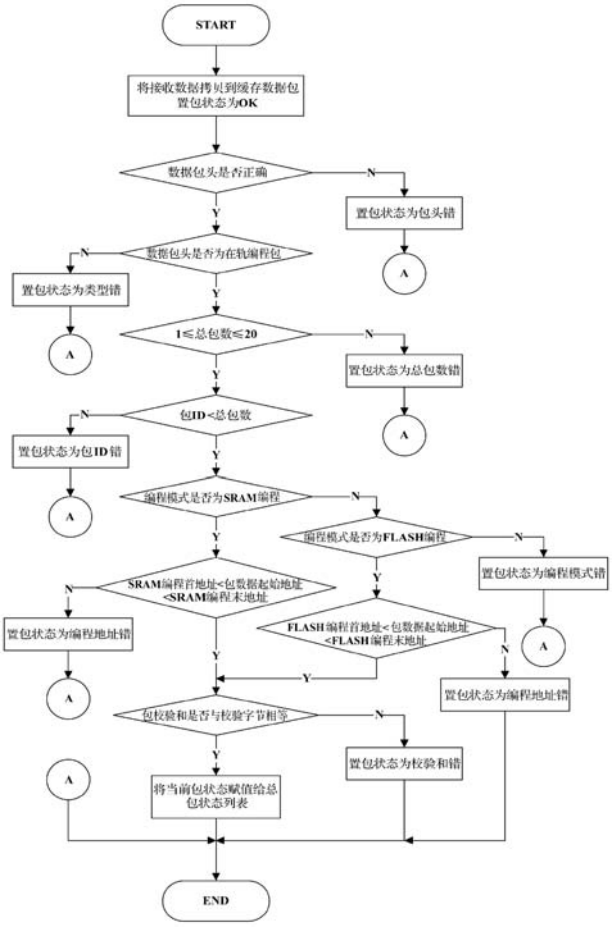

图 6　在轨编程数据包诊断示意图

当接收到在轨编程数据包后,星上软件首先对数据包内容进行解析和检查。如图 6 所示,依次对数据包头、类型、包数、ID、编程模式等信息的合法性进行诊断。若正确则将包状态置为正常,并根据数据写入地址、有效数据个数及有效数据完成数据写入。否则将包状态置为错误,并丢弃当前数据包。当所有在轨编程包均成功上注,上注状态标志则置为“上注完成”,否则置为“未上注”或“上注中”状态,通过遥测信息将包状态以及上注状态实时告知地面,用于对当前上注情况的判断。

根据此方案实现编程包上注功能的优点在于:

1) 支持最大 20 包/次的编程操作,总包数设置可根据实际需求进行修改。

2) 支持编程包的非顺序上注。

3) 支持编程包上注过程中任意包重注和更新。

以上的功能将很大程度地简化在轨编程操作的复杂性,并可完全解决上注或数据出错情况,增加了系统软件的可靠性和安全性。

3.3 编程数据使用

若编程模式为 SRAM 区编程,当确认编程包全部注入成功后,通过单地址注数包对 SRAM 区模块地址表相应模块入口地址进行修改,程序将根据该模块新的入口地址完成调用,从而实现模块替换的功能。

若编程模式为 FLASH 区编程,当确认编程包全部注入成功后,首先通过单地址注数包对 FLASH 区补丁地址表相应模块入口地址进行写入,然后利用“SRAM 区地址表更新”指令,修改 SRAM 区地址表内相应模块的入口地址,从而实现模块替换功能。

3.4 功能设计特点

首先,在轨编程功能的相关操作和使用对 FLASH 区原有的星上软件不存在任何更改和影响。一旦编程数据对卫星正常运行产生了影响,可通过指令或复位实现软件状态的快速恢复,保证了星上软件的安全性。

其次,此方案对被替换模块没有任何要求,无需在模块链接时预留多余空间,提高了空间利用率,简化软件设计。对替换模块规模基本没有限制,编程功能的使用方便、简洁。

再次,根据实际卫星软件的规模和功能,此方案中如模块地址表大小、上注最大包数等配置参数可进行灵活调整,具有较强的适用性和兼容性。

最后,此方案不仅适用于基于 SRAM + FLASH 存储构架卫星系统的在轨编程功能实现,也可以移植到基于 SRAM + EEPROM 或 SRAM + MRAM 存储构架的星载系统软件中完成此功能实现。

4 结 语

在轨编程功能是卫星软件在轨故障修复、功能扩充更新的最重要的手段。本文针对微小卫星系统软件提出一种较为完善的在轨编程功能实现方案,采用最小配置的存储构架,最大程度地满足星载软件对任意模块、任意区域编程的功能要求。同时,该方案对软件模块替换和恢复之间的切换具有快速性和易操作性,提高了卫星软件功能正常运行的可靠性和安全性,很大程度地提升了卫星在轨使用寿命。

参 考 文 献

[1] 吴海超,栾家辉,张亮,等. 航天器综合电子在轨重构容错技术研究[J]. 航天器工程,2016, 25(2):120 - 126.

[2] 董振辉,王向晖,穆强,等. 高分三号卫星中央单元多分区引导的设计与验证[J]. 航天器工程,2017,26(6):126 - 131.

[3] 夏鲁瑞, 肖龙龙, 李纪莲. 基于软件重构的小卫星在轨健康管理方法[C]//中国空天安全会议. 2017.

[4] 史江博, 郝鑫. 基于 FPGA 的小卫星通信系统在轨可重构技术研究[J]. 遥测遥控, 2017, 38(6):40 - 43.

[5] 朱虹, 王海燕. 一种星载软件在轨编程功能的设计和实现技术[J]. 上海航天, 2004, 21(1):26 - 31.

[6] 郭勇, 朱宏明, 贺彦博. 基于 ADA 语言的星载软件在轨编程技术研究[J]. 电脑知识与技术, 2008,4(7):1641 - 1644.

[7] 庞波, 郝维宁, 张文峰,等. 一种 SRAM-FPGA 在轨重构的工程实现方案[J]. 航天器工程, 2017, 26(5):51 - 56.

[8] 李振松, 杜建伟, 党纪红. 基于 DSP 平台的航天器软件在轨维护实现方法研究[J]. 空间控制技术与应用, 2017,43(6): 61 - 66.

[9] 吕敏, 张国柱, 董晋芳,等. 载荷软件可重构的空间自主飞行器内核软件设计方法[J]. 上海航天, 2015, 32(4): 54 - 58.

[10] Xu W, Piao Y. Bootstrap Loader Design of Aerospace Payload Controller Based on TSC695F[C]//Second International Conference on Computational Intelligence and Natural Computing. 2010:60 - 64.

[11] 李雁斌, 吉峰, 黄勇. 一种星载 DSP 软件的在轨编程方法[J]. 制导与引信, 2011, 32(4):37 - 41.