文章编号：2096-1472(2019)-04-33-04
DOI:10.19644/j.cnki.issn2096-1472.2019.04.009

空间飞行器主控计算机软件架构研究

贺彦博[1]，徐晓辉[2]，彭立章[1]，徐 峰[1]
(1.上海宇航系统工程研究所，上海 201109；
2.上海航天技术研究院，上海 201109)

摘 要：空间飞行器研制正在由专用化向通用化方向转型，因此必须改变当前为每个型号重新开发业务软件的现状。本文阐述了层次化软件基础平台的设计思想，引入国际标准，结合空间飞行器业务需求建立了可通用的主控计算机软件架构。

关键词：空间飞行器；飞行软件；软件架构
中图分类号：TP311.1 **文献标识码**：A

Research on Software Architecture of Spacecraft Main Control Computer

HE Yanbo[1],XU Xiaohui[2],PENG Lizhang[1],XU Feng[1]
(1.*Aerospace System Engineering Shanghai*,*Shanghai* 201109,*China*;
2.*Shanghai Academy of Spaceflight Technology*,*Shanghai* 201109,*China*)

Abstract: Spacecraft development is transforming from specialization to generalization,so the current status of software redevelopment for each project must be changed.This paper describes the design hierarchy based software platform,introduces international standards,establishes a main control computer software architecture that can be generalized in combination with the requirements of spacecraft industry.

Keywords:spacecraft;flight software;software architecture

1 引言(Introduction)

随着航天科技的持续发展，航天工业部门面临日益增长的研制压力。相比以往，研制任务更加繁重而研制周期不断缩短，同时客户又对航天器任务提出了更高的可扩展性要求。包括美国航空航天局和欧空局在内的主要国际航天组织均面临空间任务愈加复杂、软件规模持续增长的情况。以欧空局为例，该组织管辖下的空间任务飞行器主控计算机软件的规模在过去的30多年里持续增长，如表1所示。

表1 欧空局航天器软件规模增长情况

Tab.1 Growth in the size of ESA spacecraft software

发射时间	型号	软件规模
1983年	Exosat科学卫星	8k字节
1995年	SOHO太阳观测卫星	128k字节
2004年	彗星探测卫星罗塞塔	17万行，2M字节
2008年	ATV飞船	65万行，8M字节

软件规模的持续增长使得欧空局面临较大的研制压力，于是欧空局提出了空间航电开放接口体系(SAVOIR)作为空间电子设备的参考体系，该参考体系提出如下软件工程要求[1]：

(a)软件生产效率

缩短软件开发时间，节省软件确认和验证消耗，避免重复开发，提高效费比，保持并提高软件产品质量。

(b)增加反应性

缓解项目后期需求变动带来的冲击，简化并协调故障检测、隔离和恢复(FDIR)。

(c)增加灵活性

支持不同类型的系统集成策略，支持工业界策略和商用标准，支持多软件供应商策略，支持分发活动，以及未来其他需求。

满足上述要求的关键是软件构件的可重用性：

(a)体系结构重用：标准化、可组装、功能可组合的组件模型。

(b)功能重用：业务的标准化及软件组件的可重用性。

以往针对特定任务目标重新研制专用飞行软件的方式使得空间飞行器的成本高昂，因此亟待改变研制模式，在未来用途广泛、性能先进而价格可承受的通用空间飞行器基础软件平台将是研制的主流方向。空间飞行器基础软件平台需在各个型号中复用，在轨运行过程中也需要持续拓展飞行任务，因此要求软件基础平台具备架构易复用、模块易维护、功能易扩展、在轨易重构和研制成本可负担的特性。

2 当前面临的问题(Current problems)

对于空间高可靠性计算机系统而言，特别是对器件中的各种存储单元，单粒子效应是影响并制约系统可靠性及其使

用寿命的主要因素。

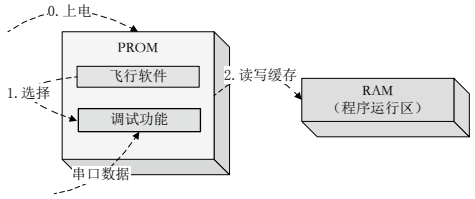

图1 基于PROM的软件部署方案

Fig.1 Software deployment scenarios based on PROM

在传统的飞行器中，主控计算机一般使用具有抗辐射指标的可编程只读存储器(PROM)、SRAM的硬件配置。由于PROM在抗单粒子方面具有高可靠性，飞行软件一般在PROM中运行，而将SRAM作为程序运行的缓存区。在默认流程中系统自动执行飞行软件业务流程以完成系统主要功能，当需要进行软件调试时，则启动调试功能从串口加载测试软件。

该模式在行业内广泛应用，技术成熟，争议最少，但该模式具有明显的缺陷。具体如表2所示。

表2 PROM芯片特性对软件的影响

Tab.2 The influence of PROM chip characteristics on software

序号	PROM芯片特性	对航天嵌入式软件的影响
1	非常昂贵、单片容量小	计算机上可配备存储空间有限
2	无法在星载计算机上在线烧写程序	软件嵌入流程复杂
3	在写入一次程序后无法修改	研制阶段末期如果出现程序错误，无法更改，无法进行迭代改进
4	抗单粒子翻转能力良好	软件设计中无须专门针对PROM做防范单粒子效应的特殊设计

图2 常用星载软件架构

Fig.2 Common satellite software architecture

现阶段空间飞行器中，大多将软件划分为设备管理软件和应用软件两个层次，在应用软件层并未做更具体的分层设计，该模式适用于针对专用装备研制专用软件的任务，易于硬件研制单位与总体设计单位分工协作。

目前已存在部分新型号将软件划分为设备访问层、服务组件层和应用任务层，具备了现代大型软件架构的雏形，但缺乏参考标准和操作系统支持，层次划分不够清晰、业务软件组件间的耦合仍然较多、软件复用率较低，若面临新的业务需求仍会引发大规模代码变更。

传统的软件设计方法在当下型号研制工作中仍然具有参考价值，但新型的多用途空间飞行器对软件提出了更高的可复用性和可维护性要求，因此亟待将更先进的软件技术投入工程应用。需解决的问题如下所示：

(a)降低软件维护的代价

目前，飞行器研制过程中软件部署流程复杂且固化后难以更改，导致软件研制成本居高不下、软件难以快速迭代，进而影响软件产品的质量；若在飞行器型号正样阶段软件固化后再次发现程序缺陷，由此带来的计算机下器处理会导致型号进度受到严重影响；处于在轨飞行阶段时进行大规模软件升级较为困难，也会极大制约飞行器空间任务拓展能力。

(b)建立软件参考架构

现阶段软件架构层次划分不清、模块间耦合过多、各型号软件架构差异过大，导致软件复用率较低，研发效率不高，研制成本居高不下。欧空局、NASA等机构的软件参考架构难以直接在我国工业部门的软件研制中落地，因此需要根据具体业务有针对性的引入行业标准，并设计适合于我国空间飞行器需求的软件参考架构。

3 软件参考体系设计(Software reference architecture)

3.1 部署与运行

为解决软件在研和在轨过程中维护代价过高的问题，要求空间飞行器主控计算机平台配置的存储硬件为PROM、FLASH/EEPROM和SRAM三者结合的方式。相比传统模式，由于FLASH/EEPROM的抗单粒子性能远不及PROM，在该新模式下如何应对单粒子效应的影响是较为重要的问题。另外，基于FLASH/EEPROM芯片的固有特性，通过设计在线烧写软件功能，使得系统具备了更加灵活的软件部署能力。芯片特性对软件的影响情况如表3所示。

表3 FLAHS/EEPROM芯片特性对软件的影响

Tab.3 The influence of FLAHS/EEPROM chip characteristics on software

序号	FLASH/EEPROM芯片特性	对航天嵌入式软件的影响
1	价格相对较低、单片容量较大	增大了星载软件可用存储空间
2	支持在星载计算机上在线烧写程序	可在计算机不从整器取下、不拆开计算机的情况下随时在线更新星载软件
3	可多次擦写	可多次修改星载软件，方便软件迭代升级
4	抗单粒子翻转能力相对PROM较弱	软件设计中需专门针对FLASH/EEPROM设计防范单粒子效应的功能模块，需对飞行软件进行三份冗余备份

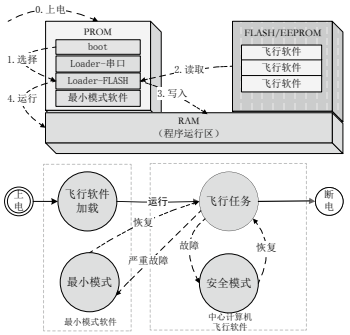

图3 基于FLASH/EEPROM的软件部署与运行方案

Fig.3 Software deployment and operation scheme based on FLASH/EEPROM

基于该硬件配置，软件系统在PROM中部署系统bootloader(启动、加载)软件和最小模式软件，在FLASH/EEPROM中部署飞行软件。其中boot功能实现计算机资源的初始化，loader功能实现从串口或从FLASH/EEPROM加载飞行软件两种运行模式，最小模式软件用以进行紧急故障应对和大规模软件在轨重构。

最小模式软件可实现系统在轨应急状况下的安全处置，并支持在轨和地面研制过程中对软件进行全部更新。在轨期间，当系统出现严重故障或者需要大规模修改时，将软件切换至最小模式，该模式以系统最简配置保证平台的姿态、能源安全，建立应急通信通道和故障反馈机制，使得飞行器具备通过地面干预而恢复正常工作状态的能力。

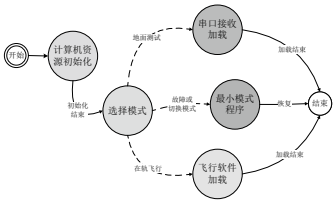

图4 启动加载软件工作模式

Fig.4 Bootloader software working mode

启动加载软件运行后通过倒计时等待用户选择加载对象，若超过倒计时无操作则自动加载飞行软件，工作模式如下所示：

(a)默认模式

在轨期间飞行器平台上电或复位后，首先从PROM运行启动加载软件，而后自动以三取二方式加载FLASH/EEPROM中的飞行软件到RAM中，并跳转到RAM中开始执行飞行软件功能。

(b)故障模式

当系统发生严重故障或需要大规模重构时可切换进入最小模式软件，在保障系统能源、温控与姿态安全的基础上进行故障监测和在轨软件重构。

(c)调试模式

在地面研制过程中，可在倒计时时刻选择串口接收加载功能，用于对FLASH/EEPROM中的程序和数据进行重新编程。

3.2 软件架构设计

软件系统实现飞行器核心管理控制，整合程控、导航、姿轨控、机构控制、遥控遥测、能源管理、温度控制、健康安全维护等功能。

应用实时操作系统，建立层次化、组件化和标准化的开放式软件架构，提高产品的灵活性、减小维护升级的代价，实现系统的灵活管理控制，为核心软件组件及飞行任务的持续革新提供良好的基础平台。

3.2.1 软件层次设计

软件设计方式由以往面向系统硬件资源处理的设计方式，转而面向基于系统需求的功能组织[2]；软件设计中将设备面与应用面分离，以设备面实现硬件资源的访问处理，以应用面实现系统平台的灵活功能定义。

设计遵循策略与机制分离的原则，以软件定义飞行器为基本设计思想，建立基于分层体系结构的开放式软件架构。

(a)策略与机制分离

软件设计中将平台的管理和控制功能划分为飞行任务策略和平台基础机制两部分，加强平台基础机制的完备性和稳定性，并以此为基础构建灵活多变的飞行任务策略，同时要求任务策略具备可快速全面重构的能力。

(b)体系结构开放性

借鉴工业界软件定义网络与欧空局软件定义卫星的设计思想，确立将应用面与设备面分离的原则，以应用面重定义实现飞行任务的灵活扩展。

软件设计实现纵向层次化和横向组件化，支持设备的即插即用管理，支持因适应技术进步或任务变化而对局部或全部软件实施快速变更，使得体系结构具有良好的开放性和较强的适应性。

软件实施层次化设计，借鉴通用开放式结构(GOA)标准将各项功能分解到应用任务管理、系统服务组件和设备资源访问三个层次[3]，如图5所示。

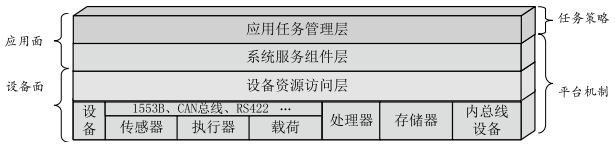

图5 软件系统层次图

Fig.5 Software system Hierarchy

(a)应用任务管理层使用实时操作系统调度各系统任务，实现平台资源的管理和试验流程的控制。

(b)系统服务组件层包含程控机制、平台内务管理、姿态轨道控制、机构控制、安全可靠性支持和应用支持库等，为任务系统提供基础应用组件支持。

(c)设备资源访问层管理平台硬件资源并提供平台状态信息和平台命令执行的操作接口。

3.2.2 软件多维度解耦及可维护性设计

主控计算机软件实现操作系统内核与应用软件、控制面与设备面、应用软件与应用软件之间等多个层次和纬度的解耦，如图6所示。

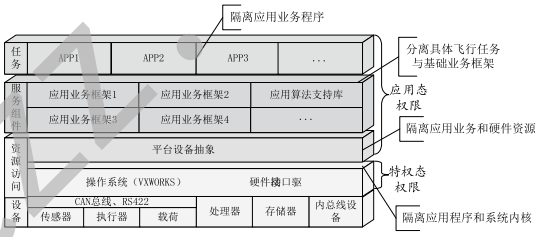

图6 多维度解耦示意图

Fig.6 Multidimensional decoupling

(a)建立平台设备抽象层以将应用业务与设备资源分离，并实现应用面的通用化和可维护性。

(b)将操作系统内核与应用软件分离编译[4]，并启用权限分级机制。

(c)应用RTP实时进程机制[4]，实现应用软件之间的空间隔离。

(d)将基础业务框架实现为动态链接库，以供各个应用程序调用。

(e)应用SOIS星上接口业务标准，实现设备与平台设备管理软件的解耦。

飞行软件将各应用业务作为独立的应用程序，操作系统实时进程(RTP)机制保证了任务调度的实时性、存储空间与时间资源的确定性[4]。内核与应用软件的隔离保证了应用程序的崩溃不影响内核的稳定性，应用软件之间的隔离保证系统的整体稳定性，具体飞行任务与基础业务框架的分离使得软件基础平台的可复用程度得以提高。

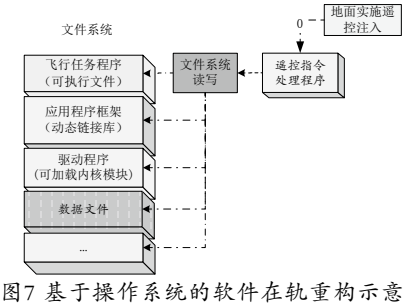

图7 基于操作系统的软件在轨重构示意图

Fig.7 Operating system-based refactoring software in orbit

系统基于实时操作系统内核系统调用和用户层软件组件库，支持应用软件和基础业务框架的独立开发和调试，并支持通过安装不同的应用程序实现地面快速集成和部署。在轨飞行期间，通过增加、删除或更改应用程序和动态链接库以实现飞行任务的快速重构。

3.2.3 行业标准参考

可通过应用行业标准降低研制成本和提高产品质量，可参照的行业标准或协议如表4所示。

表4 可参照标准/协议表

Tab.4 Reference standard/protocol table

序号	应用目标	标准/协议
1	建立整体软件框架	ESA-SAVIOR软件参考体系[1]
2	飞行器空空通信，以及天地之间的信息互联互通	高级在轨系统协议(CCSDS-AOS)[5]
3	飞行器内设备之间信息互联互通	星上接口业务协议(CCSDS- SOIS)[6]
4	实现内务管理业务，以及天地操控业务的标准化	欧空局包协议ESA-PUS[7]
5	实现应用软件与操作系统之间访问接口的通用化	NASA操作系统接口封装协议

3.2.4 软件架构设计

依据软件协同设计思想，通过纵向分层和横向组件化设计，建立资源访问、服务组件和系统任务等三个软件层面，制定各层次和各组件的标准业务及接口规范，如图8所示。

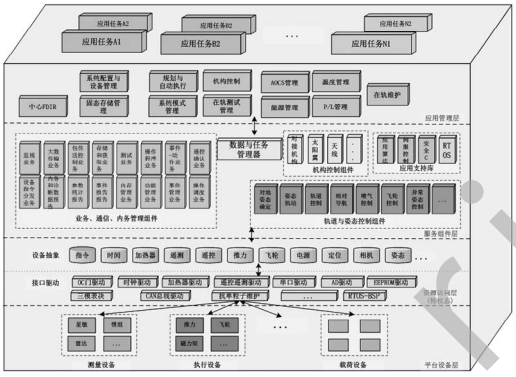

图8 软件架构图

Fig.8 Software architecture diagram

各层次功能划分如下所示：

(a)资源访问层——设备层和驱动层

设备层由构成飞行器的各种硬件设备构成。除处理器、存储器、时基、通讯接口、看门狗等计算机资源外，还包括通过总线连接的推进、载荷、电源、姿轨控传感器与执行器等飞行器设备。

驱动层包含了实时操作系统和各类设备驱动，为上层应用提供访问底层设备的API接口、三模输出数据表决和多任务管理机制等支持。

(b)资源访问层——平台设备抽象层

抽象层根据平台敏感器、执行器等设备的不同类别，对驱动层提供的操作接口进行封装，向上提供设备访问的标准接口，屏蔽底层软件实现细节，屏蔽硬件设备的细节差异，使得服务组件层的设计不至过度耦合平台设备特殊性。

通过平台设备抽象设计，实现了上层软件与设备特殊性的解耦，降低对硬件平台的依赖，最大限度保证软件的可复用性，推动未来发展中平台设备功能与接口的标准化。

(c)服务组件层

服务层包含飞行器管理控制领域常用的应用组件，为任务系统提供基础支撑。业务、通信和内务管理组件参照欧空局星地操控基础协议建立监视业务、测试业务和事件报告业务等16个子服务[7]，利用遥控包和遥测源包监视和控制星上各分系统，以及有效载荷；轨道与姿态控制组件包括导航、姿控和轨控等常用基础算法模块；机构控制组件包括天线、太阳翼等常用机构控制算法库；应用支持库包括常用数学库、安全C语言库等；数据与任务管理器提供整器状态数据的集中管理、飞行器设备状态采集和指令分发等基础管理。

(d)应用任务层

任务层包括了系统配置与设备管理、规划与自动执行、在轨测试管理、系统模式管理、AOCS管理、能源管理、中心FDIR和固态存储管理等飞行器顶层功能。

其中系统配置与设备管理任务实现飞行器设备配置的动态管理；系统模式管理任务负责飞行器在不同硬件配置情况或不同飞行阶段时工作模式的预置和切换；规划与自动执行任务负责飞行器飞行任务的子事件规划、动作执行和反馈状态监控等；在轨测试管理任务负责整器的自动测试与状态报告汇总下行功能；中心FDIR负责故障的自主诊断、隔离和处理功能；在轨维护模块负责软件的在轨重构功能[4]。

4 应用情况(Application situation)

经实践检验，基于FLASH/EEPROM的软件部署与运行模式可有效增强软件研制过程中的可维护能力和在轨飞行过程中的可重构能力，大大提高了型号软件研制的效率和质量。但对于需要长期留轨执行任务的空间飞行器，该模式在复杂空间环境中的可靠性和安全性仍有待进一步验证。

引入工业界技术成熟的操作系统可解决软件研制中目前存在的业务间高耦合、程序难维护等问题，但由于目前星载设备中处理器性能和存储空间均严重受限，且对软件的可靠性、安全性要求较高，因此对操作系统实时进程、动态链接库和可加载内核模块等技术的应用仍处于试验验证阶段。

软件参考架构的制定使得空间飞行器主控软件的开发工作有章可循，软件模块的复用率有所提高。随着软件设计方法的改进和软件组件复用率的提高，辅助以地面研制阶段在线更新和在轨运行阶段软件任务拓展等技术手段，软件的研制效率、反应性和灵活性均有所提升，推动了型号整体研制效率的提高并有效降低了综合研发成本。但该参考框架中提及的ESA-PUS等标准协议在当前星载软件业务中的应用仍处于探索阶段，需要在对现有成熟业务模式进行充分分析的基础上，结合用户实际需求，从器地一体化设计的角度对飞行器操控模式和地面应用系统进行整体技术革新。

5 结论(Conclusion)

随着空间飞行器平台由专用化向通用化的发展，必须改变针对特定飞行器重新开发飞行软件的研制模式，对飞行软件的可维护性、可复用性和研制成本等提出了更高的要求，因此构建通用化基础软件平台具有重要意义。本文描述了空间飞行器软件基础平台的设计要点，参照国际标准，确立了层次化、组件化的设计原则，设计了可通用的新型主控计算机软件参考架构。

新型软件参考架构在实际项目中的落地仍需一个长期持续的过程，需要改进软件设计方法，并建立与之适应的技术管理体制，才能真正提高星载软件的研制质量和研制效率。

(下转第16页)