Website：ycyk.brit.com.cn

星载控制软件在轨动态重构技术研究

李亚辉[1,2]，陆 钒[2]
（1 安徽大学 合肥 230039；
2 中国科学院安徽光学精密机械研究所 合肥 230031）

摘要：为使星载控制软件可在轨动态重构，提出一种基于量子编程框架、无须操作系统支持、可实现多版本切换的星载控制软件在轨动态重构方法。在分析影响在轨动态重构关键技术基础上，从量子框架的面向对象运行机制出发来寻求软件框架对动态重构的支持；通过划分函数边界，将函数归类为内部函数和公共函数，避免了模块间的循环依赖；给出了函数向量表维护策略，并以版本号为导向实现了向量表切换。该方法在BM3803星载处理器平台进行了充分测试，结果表明：所提出的在轨重构方法系统无须停机、版本可回退且更新过程可靠。本方法占用内存小、平台依赖性弱、代码可复用性强，可推广应用至硬件资源有限的星载控制器终端。

关键词：星载软件；量子框架；在轨重构；向量表
中图分类号：TP311 **文献标志码：**A **文章编号：**2095-1000(2023)03-0024-07
DOI：10.12347/j.ycyk.20221020002
引用格式：李亚辉，陆钒. 星载控制软件在轨动态重构技术研究[J]. 遥测遥控，2023，44(3)：24–30.

Research on on-orbit dynamic reconstruction technology of on-board control software

LI Yahui[1,2]，LU Fan[2]
（1. Anhui University, Hefei 230039, China;
2. Anhui Institute of Opticsand Fine Mechanics, Chinese Academy of Sciences, Hefei 230031, China）

Abstract: In order to make the on-orbit dynamic reconfiguration of onboard control software, a new on-orbit dynamic reconfiguration method of on-board control software based on quantum programming framework is proposed, which can realize multi-version switching without operating system support. Based on the analysis of the key technologies affectingon-orbit dynamic reconfiguration, the support of software framework for dynamic reconfiguration is sought from the object-oriented operation mechanism of quantum framework. By dividing function boundaries, functions are classified into internal functions and common functions to avoid cyclic dependencies between modules. The maintenance strategy of function vector table is given, and the switch of vector scale is realized based on version number. The proposed method is fully tested on the BM3803 on-board processor platform. The results show that the system of the proposed on-orbit reconstruction method does not need to shut down, the version can be rolled back and the update process is reliable.This method has the advantages of small memory consumption, weak platform dependence and strong code reusability, so it can be applied to space-borne controller terminals with limited hardware resources.

Keywords: On-board software; Quantum framework; On-orbit reconstruction; Vector table
Citation: LI Yahui, LU Fan. Research on on-orbit dynamic reconstruction technology of on-board control software[J]. Journal of Telemetry, Tracking and Command, 2023, 44(3): 24–30.

引 言

近年来，由于小卫星具有质量轻、体积小、造价低等优点，国内各大研究所掀起了研制的热潮。随着星载控制软件的任务复杂度上升，不可避免地存在着设计缺陷，需要软件具备在轨修复软件缺陷和更新算法的能力；同时为延长卫星寿命，需要在运行后期注入一些新的功能进行在轨试验。软件在轨重构已发展成为星载控制软件的基本功能之一[1,2]。

收稿日期：2022-10-20 修回日期：2022-11-28

目前星载软件在轨重构方法主要可分为两种[3]：一是静态重构。静态重构时系统不连续运行，通过加载器将上注的目标文件加载到内存的指定位置，重启系统完成功能替换或版本切换[4-6]，该方法适用于卫星运行后期在轨试验期间批量上注新的功能，而对于在轨运行期间仅修复部分软件设计缺陷和更新算法，重启系统花费的代价较大。二是动态重构。动态重构时系统连续运行，同静态重构一样，需要将上注的目标文件加载到内存的指定位置，不同之处是重构期间系统不能停机，并且在系统运行过程中进行安全升级点检测和状态的保存与恢复等工作，进而完成功能替换或版本切换。白亮等人[7]基于国产高性能风云翼辉(AIC-OS)嵌入式操作系统实现了星载软件在轨不停机更新，这类操作系统需要特定高性能处理器的支持，如用于星务计算机主机，但不适合作为远程终端的小型星载控制软件；为减少上注数据量、实现不停机更新，汪宏浩等人[8]提出基于增量链接、可回退星载软件在轨更新方法，采用增量链接方式上注代码，并进行了内存空间分配，但更新过程中未考虑安全更新时机和状态的保存与恢复等因素，存在一定的安全隐患；为适应星载软件有限的存储资源，实现细粒度的在轨更新，史毅龙等人[9]裁剪了 VxWorks 的文件系统，设计了存储于 RAM 区的函数级更新粒度，更新前通过在函数内插入跳转指令代码，去执行新函数，新函数执行完成后回到原函数结尾处继续执行，该方法对硬件依赖性较强，不利于代码移植。

针对目前星载软件在轨重构依赖大型操作系统、停机时间长、可靠性低、更新粒度较大、硬件依赖等问题，结合星载控制软件有限的硬件资源，在分析影响动态重构的关键因素的基础上，提出了一种基于量子框架的轻量级、支持系统连续运行、版本可回退、可移植性强的星载控制软件在轨动态重构方法。

1　影响动态重构的关键因素

星载控制软件所处特殊的运行环境，为保障系统能够实现安全可靠的在轨动态重构，应在系统设计过程中解决以下问题[10]：

① 划分重构单元。受天地通信速率和过站时间等因素影响，星载控制软件在轨重构期间应尽可能地减少代码上注量，良好的模块化设计可有效减少代码的耦合度、提升代码的可复用性。重构模块应遵循以下设计原则：一个模块只负责实现一项功能，承担过多的职责将加大重构的粒度，增加代码的上注量；提高模块的扩展性，避免对原有功能修改，有利版本回退；针对接口进行编程，模块间依赖抽象而非依赖实现，星载控制软件功能的重构通过修改参数表进行配置。

② 选取安全更新点。为防止在轨重构发生在系统执行关键时期，影响载荷的安全运行，如：待重构模块执行期间进行重构可能会导致系统数据丢失；载荷姿轨控制的关键时期进行重构会影响载荷的平衡。故选取安全的更新点至关重要。为实现选取安全更新点，要求系统具备检测当前执行位置和运行状态等监视功能。然而，一般来说这个监视功能是难以实现的，所以必须要有系统架构的支持。

③ 保护断点。任务运行过程不可避免地需要中间变量和历史状态的支持，当待重构单元中存在静态变量等标志系统运行状态的元素时，重构前需要为此类单元备份变量名和变量值，以便将其赋值给重构后的对应单元，保证重构单元完成重构后不丢失历史状态和数据，保证系统的连续运行。

④ 重定位。重定位是代码链接的关键一步。对于静态链接，代码在烧录到系统内存前完成重定位，由于系统运行时函数地址固定不变，需要在系统设计之初为可重构单元预留足够的存储空间，以满足重构单元的代码扩充，易造成内存空间的浪费，静态链接方式难以实现星载控制软件的不停机重构；对于动态链接，重定位操作将会延缓到代码运行时，只需在每个任务模块内维护一组存储函数入口地址的重定位表。重构单元上注时将目标代码存储至内存空闲区域，在系统安全升级点更新重定位表即可实现在轨不停机重构。

2　星载软件在轨动态重构方法

2.1　在轨动态重构方法概述

随着软件重用技术的日臻成熟,利用事件驱动框架构造软件系统已成为软件开发的重要手段。事件驱动框架由一系列精细粒度的状态处理函数来处理事件，这些状态处理函数执行结束后很快返回主事件循环，在调用树里不需要维护上下文和程序计数器(Program Counter，PC)，方便获取系统执行位置和运行状态。而量子框架(Quantum Fr-

amework，QF)[11]正是一种基于事件驱动实时并发状态机的应用框架，尤其适用于嵌入式软件重构系统。本章从量子框架下的重构分析出发，寻求事件驱动型框架对动态重构的支持，并基于量子框架设计了动态链接系统，从而实现了星载软件在轨动态重构，其主要重构过程包括加载重构单元、检测安全更新点、状态转移和重定位。

2.2 量子框架下重构分析

应用量子框架可以简单地将整个系统分为相互独立的活动对象，每一个活动对象在量子框架中被封装成一个任务，每个活动对象中都嵌入一个状态机完成所要求的任务。活动对象不共享任何数据，它们之间唯一的通信手段是通过量子框架来进行事件实例的交换。图 1 所示是一个活动对象执行结构，它包含一个控制线程(事件循环)、一个事件队列和一个状态机。图 1 右侧为事件循环流程，事件队列中的事件被 queue.get( )取出，由 dispatch()派送到状态机进行处理。在处理完成后再次从事件队列中取出、派送事件，如此循环。

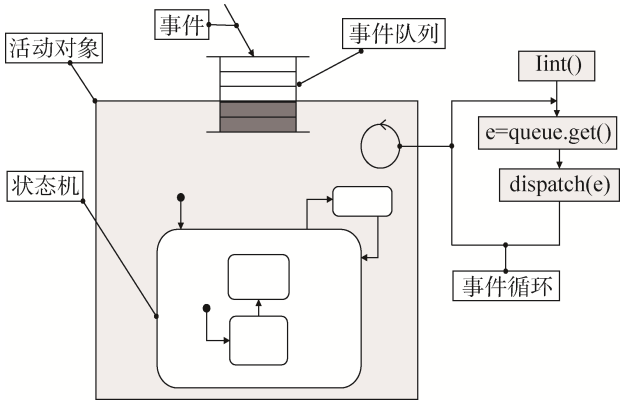

图 1 活动对象执行结构

Fig.1 Active object execution structure

为了体现量子框架在动态重构系统设计中的优势，将量子编程与传统编程进行了 4 处对比：① 轻量级框架。传统操作系统支持下的动态重构方式对存储器的容量要求比较苛刻，如 VxWorks 嵌入式操作系统，RAM 和 ROM 的占用量都在 100 K 以上，限制了动态链接技术在微小型卫星载荷领域的应用。量子编程框架为一个轻量级事件驱动型实时嵌入式软件框架，最小仅需 100 B RAM 和 2 KB ROM 便可构建响应型多任务实时并发应用[11]。② 一种 RTC(Run-To-Completion，运行至完成)事件处理机制。传统检测系统运行位置的方式是通过判断程序计数器指针 PC 是否在待重构模块的地址空间以及堆栈中是否有该模块的地址来获取系统运行位置，该方法直接访问内部寄存器，严重依赖硬件平台，不利于代码重用与移植。在量子编程模式下，活动对象以 RTC 方式处理循环队列中的事件，重定位事件处理时保证 PC 指针指向框架而不是应用程序，使检测系统运行位置更加简单。③ 发布-订阅(Publish-Subscribe)事件派发。量子框架支持使用 FIFO 和 LIFO 策略对响应任务进行直接事件发送，也支持发布-订阅模式的事件派发机制。传统的直接事件派发模式下，当一个事件发生后，发送方把事件直接发送到接收方，接收方可能是一个也可能是多个，发送方必须清楚知道每一个接收方，收发任一方的修改都会致使多方代码的调整，造成了循环依赖，在循环依赖下卸载模块时对系统安全性影响较大[12]；发布-订阅模式下，事件首先被发送至中介，其次中介把事件逐一发送给事件订阅者，活动对象由循环依赖转为单向依赖，对任务间事件传递进行了解耦，便于划分重构单元。④ 状态机编程。传统顺序式编程过程中需要设置大量的中间标志变量来保存系统的执行状态，系统工作状态难以分析。状态机编程下，系统结构清晰、逻辑完备，重构过程中便于找出安全更新点及断点保护，使重定位操作可靠性高。

2.3 动态链接系统设计

动态链接系统分为以下 5 个方面：

① 设计重构对象。量子框架基于状态机构建系统功能，状态机由状态、事件、动作三部分组成。其中状态由状态处理函数构建，图 2 为活动对象状态处理函数，事件和动作均为状态处理函数成员，动作则由若干单一功能函数组成。为有效节省内存空间，将单一功能函数根据被调用情况划分为内部函数和公共函数。内部函数被封装在当前活动对象内部，只能被当前活动对象内部状态机调用，对于被两个以上活动对象调用的函数和数据结构则是以公共函数的形式存放到公共链接库内，不同活动对象使用动态链接方式调用公共函数[13]。因此，重构对象分别为状态处理函数、内部函数以及公共函数。

② 向量表维护方法。函数调用基于访问向量表机制获取函数入口地址。图 3 为向量表结构，每个活动对象拥有一组独立向量表参数，向量表存

状态处理函数

```
QState  Bus_1553B_DataHead (DataHead *me, Qevt const *e) {
  static u8  sta = 0;                    // 历史状态变量
  switch  (e->sig)  {                    // 判断事件类型
      case Q_ENTRY_SIG: {                // 进入事件
          // 处理进入事件的动作（内部函数或公共函数）
           return Q_HANDLED();           // 事件被处理，返回框架
        }
      case A _SIG:  {                    // A 事件
          // 处理进入A事件下的动作（内部函数或公共函数）
          return Q_HANDLED();            // 事件被处理，返回框架
        }
      case  B_SIG:  {                    // B 事件
          // 处理进入B事件下的动作（内部函数或公共函数）
          return Q_TRAN(&Bus_1553B_Loader);//切换到重构加载状态函数
        }
      case  SAVESIG:  {                  // 保护现场事件
          // 处理进入保护现场事件下的动作（静态变量的保存）
          return Q_TRAN(&Bus_1553B_Loader); // 切换到重构加载状态函数
        }
      case  RECOVER_SIG:  {              // 恢复现场事件
          // 处理进入恢复现场事件下的动作（静态变量的恢复）
          return Q_TRAN(&Bus_1553B_Loader); // 切换到重构加载状态函数
        }
    }
  return Q_SUPER(&QHsm_top);             // 未知事件，交由父类处理
}
```

图 2　活动对象状态处理函数

Fig.2　Active object state handler

向量表

```
enum StaFunName{       //状态处理函数索引号
        Sta1Fun=0,
        Sta2Fun,
//      ...
        StaNFun,
        IdleFun,       //空闲函数
        StaFunNum      //状态转移函数数量
};
enum PrivateFunName{       //内部函数索引号
        Private1Fun=0,
        Private2Fun,
//      ...
        PrivateNFun,
        IdleFun,           //空闲函数
        PrivateFunNum      //内部函数数量
};
enum PubFun{      //公共链接库函数索引号
        Pub1Fun=0,
        Pub3Fun=10,
//      ...
        PubNFun=50,
        IdleFun        //空闲函数
};
const struct VECTORTABLE_PRIVATE{          //活动对象向量表结构
        u32   PubVectorTableAddr;//公共链接库向量表入口地址
        u32   StaFunc[StaFunNum];//存放函数地址
        u32   PrivateFunc[PrivateFunNum];
}FlashVector[2];      //存放两份向量表，用于版本更新
```

图 3　向量表结构

Fig.3　Vector table structure

储着活动对象内状态机的状态转移函数、内部函数的入口地址以及公共链接库中向量表的入口地址。活动对象和公共链接库中向量表各存储两份，满足星载控制软件版本切换需求。公共链接库作为主题被其他活动对象订阅，当公共链接库中的函数发生更新时，将更新后的公共链接库中向量表入口地址通过发布事件通知所有订阅者(活动对象)，使之完成向量表中的公共链接库向量表入口地址重定位。此外，为便于卫星运行后期扩展星载控制软件功能，通过在函数索引号内插入若干空闲函数索引号(IdleFun)为向量表预留一定的函数入口地址存储空间。

③ 重构函数注册与加载。重构函数注册与加载过程依赖向量参数表，向量参数表以结构体形式组织，存储着函数向量表的一些重要参数，包含向量表基地址、状态处理函数基地址、内部函数基地址以及向量表内存空间大小。在系统启动时，每个活动对象按优先级顺序将自身向量参数表注册到重构加载器，此外，重构加载器把公共链接库的向量表入口地址加载到每个活动对象向量表中。在轨重构过程中，加载器对向量表函数入口地址写入时依据以下公式定位：函数指针地址=基地址+向量表内存×(版本号%2)+函数索引号×4。

④ 实现安全更新点检测方法。在进行向量表重定位前，需要判断当前状态是否为安全更新点，原理如下：在执行关键任务时刻，对于重定位事件，设计延迟事件机制将重定位向量表事件进行延迟处理，通过改变重定位事件在循环队列中的顺序来延迟该事件，待关键任务执行完成后，此时重定位事件的执行环境为安全更新点。图4为延迟事件状态图，重定位事件到达后，先转到接收(receiving)状态中查看当前是否有事务正在接收和处理，如果无事务正在接收或处理，则重新回到空载(idle)状态进行收回(recall)并处理重定位事件；如果事务正在接收或处理时，状态转移到占用(busy)父类对重定位事件进行延迟(defer)，待事务处理完后转换到空载(idle)进行收回(recall)并处理重定位事件。

⑤ 保护和恢复现场。状态处理函数内一般含有运行状态及中间变量等信息，重构过程中需要对其进行备份、转移、还原等操作，称为保护和

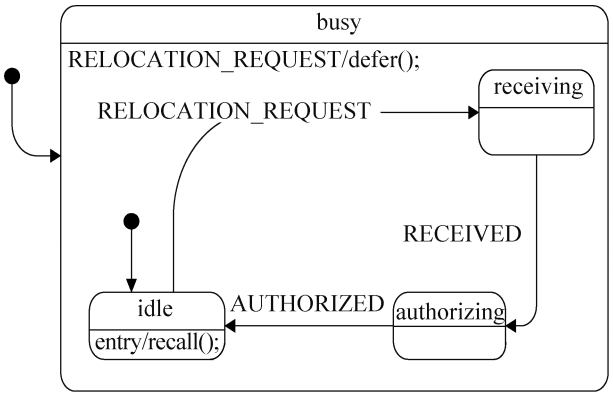

图 4　延迟事件状态图

Fig.4　Defer event status diagram

恢复现场。在需要进行保护和恢复现场的状态处理函数内设计保护现场事件和恢复现场事件的动作。重定位前后，由重构加载器分别产生保护现场事件和恢复现场事件，在重构前的状态处理函数中进行现场保护，在重构后的状态处理函数中进行现场恢复。

3 动态重构实现与试验验证

本文试验基于BM3803星载嵌入式系统硬件平台,主控CPU为国产高性能SPARC V8架构的抗辐射32位RISC处理器BM3803 MGRH；选用目前在航空航天领域广泛应用的1553B总线BU65170作为远程终端(RT)与星务计算机通信；单片FLASH型FPGA，采用ACTEL公司ProASIC3系列芯片，低功耗和内置加密技术，为BM3803与1553B总线应用通信建立安全可靠的逻辑接口，适应于1553B总线的异步通信需求；外扩32 KB ROM、512 KB RAM、1 MB EEPROM，均使用抗辐射存储器，可有效克服空间辐射环境中的单粒子效应,系统的启动代码和框架代码链接到ROM内，在轨运行期间不可擦除。

3.1 软件系统搭建

星载控制系统软件结构如图5所示，共分为三层：应用层、框架层、硬件驱动层，软件分为可重构部分和不可重构部分，图中虚线部分为可重构部分。系统软件搭建主要包含2个方面：① 开发板级支持包。在框架层开发与硬件平台相关的片级初始代码：配置BM3803的系统时钟和堆栈区；初始化异常向量表；搭建量子框架所需的时钟节拍、内存池；设置CPU的核心寄存器和控制寄存器；完成程序跳转[14]。设计系统级驱动代码：使用硬件代理模式设计与硬件相关的驱动程序，把对外设寄存器等操作转换成对标准外部接口的操作。② 开发在轨重构所需的应用层模块。应用层分别实现以下活动对象模块：总线收发活动对象负责与星务主机通信，解析1553B总线收发的数据并分发给相应活动对象；重构加载器活动对象负责接收来自总线收发活动对象的重构指令及数据，将数据解析、拼包后加固到FLASH以及执行向量表重定位等操作；公共链接库活动对象负责存储并管理公共函数；状态监测活动对象负责监控系统的实时运行状态，并实现纠错检测功能，重构过程中监视系统工作状态；串口调试活动对象负责在地面调试阶段在线调试，同时打印运行状态；测试模块活动对象负责在轨重构功能验证。

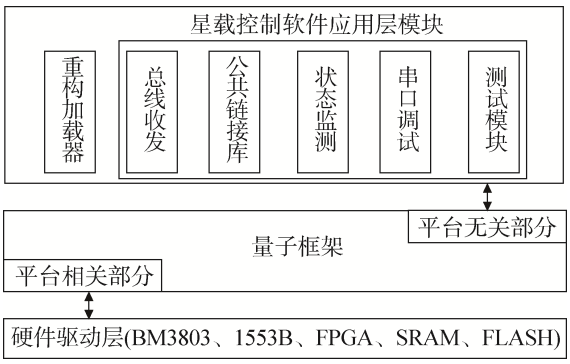

图5 星载控制系统软件结构

Fig.5 Software structure of onboard control system

3.2 函数加载与重定位

① 函数加载。代码修改时以函数为最小修改单元，修改后的代码链接到内存的空闲区域并为每个重构函数的目标代码增加函数参数表，如表1所示，将目标代码和函数参数表打包上注至星务计算机。星务计算机将重构代码通过1553B总线分包发送至星载控制软件终端的总线收发器，重构加载器读取总线收发器的重构数据包，并在系统运行非关键任务时刻执行加载函数操作，并将参数表跟随版本号一同存储在重构加载器中，用于版本回退。

② 重定位操作。函数调用时函数指针通过版本号定位所属向量表，即通过更新版本号实现重定位。图6为星载控制软件重定位流程。首先应判断重构函数类型，分别为状态处理函数、内部函数和公共函数，不同类型重构单元进行重定位操作时有不同的操作步骤。当重构单元为状态处理函数时，需要判断状态处理函数内部是否包含静态变量，包含静态变量的状态处理函数需要在更新版本号之前执行保护现场操作，并且在更新完版本号后在新状态处理函数中恢复现场，反之则直接进行版本号更新；当重构单元为内部函数时，由于RTC机制，可直接进行版本号更新；当重构单元为存储于公共链接库中的公共函数时，把更新后的公共链接库中向量表入口地址加载至各个活动对象向量表中。

3.3 试验验证

基于上述硬件平台，对所提出软件在轨动态重构方法进行试验，主要针对3种类型函数功能更

表 1　重构函数参数表

Table 1　The function parameter table of reconstructs

参数名称	数据大小/B	备注
目标主机 ID	1	1553B 可挂接 31 个远程终端，ID 范围：0X01~0X1F
函数版本号	1	支持 0~255 个版本在轨更新
所属单元	1	代表活动对象 ID，最大并发管理 63 个活动对象
函数类型	1	状态处理函数(0×01)，内部函数(0×02)，公共函数(0×03)
函数索引号	2	函数在向量表中的位置
函数入口地址	4	存储到向量表中
函数占用空间大小	2	最大支持 65 535 B 空间
代码校验值	2	函数代码的 CRC 值

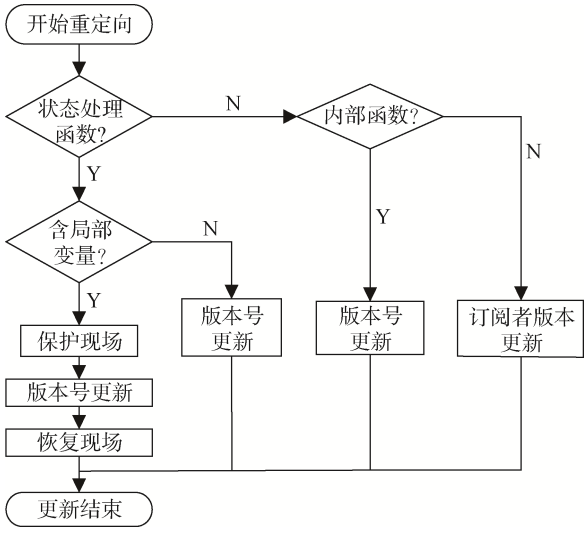

图 6　星载控制软件重定位流程

Fig.6　Relocation process of onboard control software

新效果、系统运行的连续性和版本回退功能进行测试。本次试验重点验证量子框架下的动态重构系统的可行性，使用两台 BM3803+1553B 实验板，其中一台作为星务主机，1553B 配置为总线控制器(BC)，另一台作为星载控制器，1553B 配置为远程终端(RT)。星务主机通过串口线连接电脑，通过电脑发送上注指令模拟星地链路，串口波特率为 115 200；1553B 总线严格按照星上实际在轨通信速率 1 Mbps 和 32 字的消息包进行通信。

实验在版本号 1 的基础上进行了 6 次版本修改，形成了编号为 1~7 的版本。测试阶段通过电脑串口调试界面将系统的运行状态和重构过程反馈回来。表 2 显示了 17 次更新/回退的实验结果，其中重构单元列中的‘0’表示状态处理函数，‘1’表示内部函数，‘2’表示公共函数，‘–’表示无操作，本实验每次只对单个函数进行更新测试。

表 2　软件重构执行结果

Table 2　Result of software refactoring

序号	操作	重构单元	当前版本	重构版本	重构后版本	运行状态	版本切换
1	更新	0	1	2	2	连续	成功
2	更新	0	2	3	3	连续	成功
3	更新	2	3	4	4	连续	成功
4	回退	–	4	3	3	连续	成功
5	更新	1	3	4	4	连续	成功
6	更新	2	4	5	5	连续	成功
7	回退	–	5	4	4	连续	成功
8	更新	1	4	5	5	连续	成功
9	回退	–	5	4	4	连续	成功
10	更新	1	4	5	5	连续	成功
11	更新	1	5	6	6	连续	成功
12	更新	0	6	7	7	连续	成功
13	回退	–	7	6	6	连续	成功
14	更新	1	6	7	7	连续	成功
15	回退	–	7	6	6	连续	成功
16	更新	1	6	7	7	连续	成功
17	回退	–	7	6	6	连续	成功

表 2 数据显示：经过 17 次重构操作实验，更新成功与失败次数分别为 11 和 0，回退成功与失败次数分别为 6 和 0。分析以上操作结果可知：① 当执行更新操作时，3 种重构单元均能正常执行；② 当执行版本回退操作时，均可实现单级回退；③ 不同重构单元下执行的更新和回退均不影响系统的正常运行状态，满足不停机要求。上述实验测试结果表明：本文提出的动态重构方法能够可靠地实现软件不停机更新和版本回退。

4 结束语

在轨重构技术是提高星载控制软件可靠性和可扩展性的有效手段。本文在分析在轨动态重构关键技术的基础上，基于量子框架运行机制，实现了一种无须重启、版本可回退、微内核的星载控制软件在轨重构系统，并在BM3803+1553B高可靠性硬件平台进行了充分测试。该方法解决了在轨动态重构过程中安全更新点选取难、运行数据易丢失等关键技术难题。所用框架占用内存小、平台依赖性弱、代码可复用性强，可推广应用至微内核星载控制器终端。

参考文献

[1] 白照广. 中国现代小卫星发展成就与展望[J]. 航天器工程, 2019, 28(2): 1-8.
BAI Zhaoguang. Development achievements and prospects of modern small satellites in China[J]. Spacecraft Engineering, 2019, 28(2): 1-8.

[2] JI X, LI Y, LIU G, et al. A brief review of ground and flight failures of Chinese spacecraft[J]. Progress in Aerospace Sciences, 2019, 107(5): 19-29.

[3] 吴海超, 栾家辉, 张亮, 等. 航天器综合电子系统在轨重构容错技术研究[J]. 航天器工程, 2016, 25(2): 120-126.
WU Haichao, LUAN Jiahui, ZHANG Liang, et al. Study on fault tolerance technology for on-orbit reconstruction of spacecraft integrated electronic system[J]. Spacecraft Engineering, 2016, 25(2): 120-126.

[4] 靳鑫, 徐清华, 王慧泉, 等. 星载软件的语句级高效更新方法[J/OL]. 哈尔滨工业大学学报: 2022, 54(12): 38-45 [2022-10-19].
JIN Xin, XU Qinghua, WANG Huiquan, et al. A Statement level efficient update method for spaceborne software[J/OL]. Journal of Harbin Institute of Technology: 2022, 54(12): 38-45 [2022-10-19].

[5] 钱方亮, 林荣锋, 周宇, 等. 一种基于微小卫星系统软件在轨编程功能的设计方法[J]. 计算机应用与软件, 2018, 35(12): 16-20.
QIAN Fangliang, LIN Rongfeng, ZHOU Yu, et al. A design method based on on-orbit programming function of micro-satellite system software[J]. Computer Applications and Software, 2018, 35(12): 16-20.

[6] 李磊霞, 王宇, 林宝军, 等. 基于宏定义动态链接的模块化星载软件升级方法研究[J]. 空间科学学报, 2010, 30(2): 180-184.
LI Leixia, WANG Yu, LIN Baojun, et al. Research on modular spaceborne software upgrade method based on macro definition dynamic link[J]. Chinese Journal of Space Science, 2010, 30(2): 180-184.

[7] 白亮, 邱源, 韦杰, 等. 基于动态库的星载软件可重构设计与实现[J]. 上海航天(中英文), 2021, 38(4): 84-91.
BAI Liang, QIU Yuan, WEI Jie, et al. Design and implementation of spaceborne software reconfigurable based on dynamic library[J]. Shanghai Aerospace (Chinese and English), 2021, 38(4): 84-91.

[8] 汪宏浩, 王慧泉, 金仲和. 基于增量链接的可回退星载软件在轨更新方法[J]. 浙江大学学报(工学版), 2015, 49(4): 724-731.
WANG Honghao, WANG Huiquan, JIN Zhonghe. On-orbit update method of backtractable spaceborne software based on incremental link[J]. Journal of Zhejiang University (Engineering Science), 2015, 49(4): 724-731.

[9] 史毅龙, 薛长斌. 基于“龙芯”的VxWorks系统函数在轨更新研究[J]. 电子设计工程, 2015, 23(21): 106-109.
SHI Yilong, XUE Changbin. Based on VxWorks function of the "dragon chip" in-orbit renewal research[J]. Journal of electronic design engineering, 2015, 23(21): 106-109.

[10] 鲍春健, 吴俊敏, 许胤龙, 等. 基于组件的动态软件更新[J]. 计算机应用, 2006(8): 1909-1911.
BAO Chunjian, WU Junmin, XU Yinlong, et al. Component-based dynamic software update[J]. Computer Applications, 2006(8): 1909-1911.

[11] MIRO Samek. Pracrical UML statecharrs in C/C++, Second Edition[M]. Burlington: Newnes, 2008.

[12] 刘赟, 左小川. 嵌入式软件在线升级系统的设计与实现[J]. 计算机测量与控制, 2015, 23(4): 1425-1427.
LIU Yun, ZUO Xiaochuan. The design and implementation of embedded software online upgrade system[J]. Computer measurement and control, 2015, 23(4): 1425-1427.

[13] 王利涛. 嵌入式C语言自我修养: 从芯片、编译器到操作系统[M]. 北京: 电子工业出版社, 2021.

[14] 韩春慧, 王煜, 黄书华, 等. 基于BM3803的1553B总线通信软件设计[J]. 中国空间科学技术, 2019, 39(5): 61-68.
HAN Chunhui, WANG Yu, HUANG Shuhua, et al. BM3803 based on 1553B bus communication software-design[J]. China's Space Science and Technology, 2019, 39(5): 61-68.

[作者简介]

李亚辉　1996年生，硕士研究生，主要研究方向为星载嵌入式开发。

陆　钒　1976年生，副研究员，硕士生导师，主要研究方向为星载嵌入式开发。

(本文编辑：潘三英)