计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(1):72−77 [doi: 10.15888/j.cnki.csa.006148]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

高可靠的计算机模块引导软件①

周鹏举, 倪　明, 施华君, 杨三岭

(中国电子科技集团公司第三十二研究所, 上海 201800)
通讯作者: 周鹏举, E-mail: 815324919@qq.com

摘　要: 在航天软件领域, 为适应相关软件的快速发展与迭代, 需要一种引导软件, 使其能够对不同应用程序进行重构并保证其高可靠性. 本文基于 SPARC 架构设计并实现了一种引导软件, 该引导软件既可以根据引导标识自动引导某个程序, 又可以在地面指令控制下进行应用程序重构、引导等. 同时, 采用三冗余架构、反弹墙、EDAC 保护等可靠性措施, 确保运行过程中的大多数软件错误能够得到恢复. 最终在计算机模块上进行测试, 测试结果达到了预期目的.

关键词: 引导软件; 重构; 三冗余架构; 高可靠性

引用格式: 周鹏举,倪明,施华君,杨三岭.高可靠的计算机模块引导软件.计算机系统应用,2018,27(1):72–77. http://www.c-s-a.org.cn/1003-3254/6148.html

High Reliability Computer Module Bootloader

ZHOU Peng-Ju, NI Ming, SHI Hua-Jun, YANG San-Ling

(The 32nd Research Institute of China Electronics Technology Group Corporation, Shanghai 201800, China)

Abstract: In the field of aerospace applications, a bootloader, aiming to satisfy the rapid development and iteration of the applications, is needed to reconstruct different applications and ensure its high reliability. Based on the SPARC architecture, this study designs and implements a booloader, which cannot only boot applications automatically according to the bootflag, but also can reconstruct or boot a special application under the control of the ground command. Also, the three redundant architecture, rebound wall, EDAC protection, and other reliability measures have been taken to ensure that the most software failure can be restored. Finally, we test the bootloader on the computer module, and the results have achieved the desired purpose.

Key words: bootloader; reconstruct; three redundant architecture; high reliability

自第一颗人造卫星进入太空以来, 航天技术取得了迅猛的发展, 现已广泛应用于国民生活的方方面面. 我国经过 40 多年的艰苦努力, 在航天方面取得了一系列重大成就. 随着计算机模块运算能力的提升和任务的增多, 计算机模块应用程序呈现出多样化和复杂化的发展趋势. 快速迭代的软件需求, 要求设计一种计算机模块引导软件 (以下简称引导软件), 使其能够对应用程序进行重构和引导等, 并且为适应复杂的空间环境, 需要采用一定措施保证其高可靠性.

本文针对上述问题, 设计与实现了一种基于 SPARC 架构的高可靠引导软件. 该引导软件的主要功能如下:

1) 自检功能, 确保引导软件本身正确.

2) 硬件检测功能, 确保硬件状态正确.

3) 三冗余架构、反弹墙、EDAC 保护等多种高可靠性措施, 确保软件的正确运行.

4) 重构功能, 便于不同应用程序的在线固化, 缩短研发周期.

5) 多种引导方式, 便于对应用程序的调试.

① 收稿时间: 2017-04-07; 修改时间: 2017-04-26; 采用时间: 2017-05-02; csa 在线出版时间: 2017-12-22

1 硬件环境概述

1.1 计算机模块与调试环境

本实验所用的计算机模块原理框图如图 1 所示, CPU 使用基于 SPARC 架构的 AT697F, 主要包含 RAM、PROM、EEPROM 三种存储芯片, RAM 是计算机模块的内存区域, 引导软件烧写于 PROM1(实验过程中由 EEPROM 代替) 中, 自动引导的应用程序 (以下简称应用程序) 烧写于 PROM2(实验过程中由 EEPROM 代替) 中, 可重构的应用程序 (以下简称重构程序) 固化于 EEPROM 中.

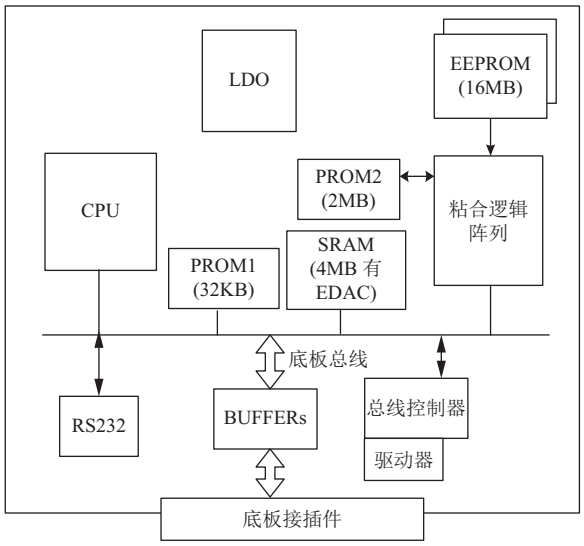

图 1 计算机模块原理框图

计算机模块通过 RS232 与外界通信. 计算机模块与计算机的通信示意图如图 2 所示, 计算机使用调试环境 VDS 通过仿真器 LEON 将引导软件下载到计算机模块中, 计算机中的地面服务程序 (界面见图 10) 通过 RS232 串口与计算机模块进行通信.

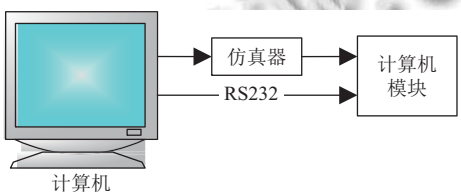

图 2 计算机与计算机模块通信图

1.2 CPU 简介

该实验采用的 CPU 是 Atmel 公司的 AT697F. AT697F 基于 SPARC V8 架构设计, 采用 RISC 精简指令集, 具有 8 个寄存器堆窗口、5 级流水线, 使用容错设计技术, 具有软件可控的省电工作模式, 是一种高性能低功耗的 32 位嵌入式处理器.

AT697F 主频最高可达 100MHz, 定点运算性能 86MIPS, 浮点运算性能 23MFLOPS. 正常工作温度范围广, 抗辐射能力强. 该 CPU 的各项指标能够满足卫星在轨运行期间的各项性能要求.

1.3 计算机模块存储空间

AT697F 具有 32 根地址总线, 可寻址 4 GB 的地址空间. 该计算机模块采用统一编址方式, 地址空间分配如表 1 所示.

表 1 计算机模块地址空间

序号	地址范围	大小	说明
1	0x00000000~0x00007FFF	32 KB	PROM1地址空间
2	0x00008000~0x00207FFF	2 MB	PROM2地址空间
3	0x20000000~0x20FFFFFF	16 MB	EEPROM地址空间
4	0x23000000~0x230000FF	256 B	I/O地址空间
5	0x40000000~0x403FFFFF	4 MB	SRAM地址空间

2 引导软件设计方案

2.1 引导软件的模块划分

本引导软件的设计方案如图 3 所示, 主要分为 6 个模块.

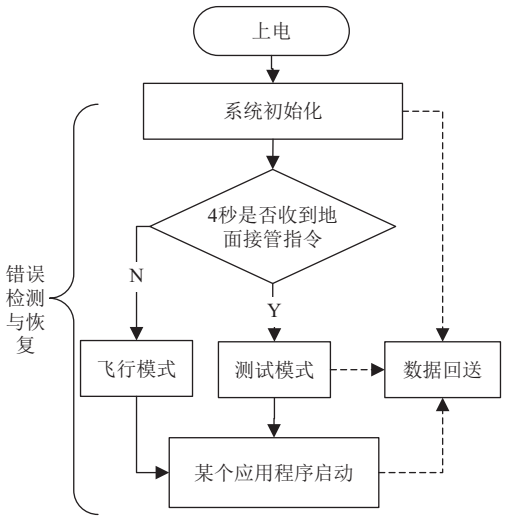

图 3 引导软件设计方案

系统初始化模块用于系统初始化和引导程序的搬移和跳转. 系统初始化主要包括 CPU、外围器件, 陷阱/中断、堆栈等的初始化. 引导程序的搬移和跳转用于引导程序自身的转移.

系统初始化完成后, 引导软件等待 4 s, 若 4 s 内未收到地面接管指令则进入飞行模式模块, 否则进入测试模式模块. 飞行模式模块是系统的自动管理模块, 它

根据引导标志自动完成应用程序或重构程序的引导; 测试模式模块根据地面指令执行相应操作, 如重构程序上注、固化等, 并最终引导一个应用程序的启动.

数据回送模块用于引导程序运行过程中状态的回送, 如重构程序校验和的回送等.

错误检测与恢复主要包括 RAM 正确性检测、程序比特位正确性检测、系统运行正确性检测和重构程序三冗余架构设计等, 若检测到错误, 则尝试进行恢复.

本文主要对系统初始化、飞行模式、测试模式和错误检测与恢复 4 个模块进行论述.

2.2 引导软件的目录结构

该引导软件主要由 trap.s、setup.s、xxmain.c 等 3 个文件组成, trap.s 主要用于设置陷阱表和中断临界区, setup.s 主要包含初始化和错误检测与恢复两个模块, xxmain.c 完成引导软件的引导测试功能, 即完成飞行模式、测试模式和程序启动三个模块.

3 引导软件的实现

3.1 系统初始化

系统初始化主要包括硬件初始化和引导程序的搬移和跳转.

3.1.1 硬件初始化

AT697F 共有 8 个全局通用寄存器和 8 个寄存器窗口, 每个寄存器窗口对应一组通用寄存器. 初始化时对上述通用寄存器全部清 0, 对控制寄存器进行配置, 主要配置当前寄存器窗口、无效窗口、cache、内存等.

中断是现代处理器的重要组成部分, 本引导软件在系统初始化时完成对中断的配置. 中断服务函数在 xxmain.c 中完成, 临界区的进入/退出和中断向量表在 trap.s 中, 中断配置在 setup.s 中完成.

AT697F 的中断控制在多个寄存器中完成. 主要是设置中断屏蔽级别 PSR.pil 为 0, 设置异常基址标志寄存器 TBR 为中断向量表的入口地址, 其它与中断相关的寄存器如中断级别/屏蔽寄存器等都设为 0. 当程序跳转至主函数前置 PSR.ET 为 1, 打开中断.

因 AT697F 共有 8 个寄存器窗口, 故需为每个窗口设置一个堆栈, 该引导软件的堆栈栈底位于 SRAM 的 0x400f0000 处, 代码为:

PROVIDE (__stack = 0x400f0000);

每个窗口对应一段栈空间, 栈指针 %sp 向下增长.

3.1.2 引导程序的搬移和跳转

引导程序固化于 PROM 中, 运行时将自身搬移到 RAM 中, 自检正确后跳转至引导软件的 main 主函数.

3.2 错误检测与恢复

因太空复杂的环境对程序的正确运行构成极大威胁, 本引导软件为保证程序的正确运行采取如下高可靠性措施, 在文件 setup.s 中主要完成 RAM 正确性检测, EDAC 保护, 反弹墙保护, 看门狗保护等; 在文件 xxmain.c 中主要完成三冗余架构设计. 其中 RAM 正确性检测属于硬件正确性检测, EDAC 保护属于比特位正确性检测与恢复, 反弹墙保护、看门狗保护属于系统运行正确性检测, 三冗余架构属于错误检测与错误恢复, 将在 3.4.2 节论述.

1) RAM 错误检测

RAM 的起始地址是 0x40000000, 错误检测的方法是在 RAM 的地址空间中, 依次向每个字中写入 0x0 并读出, 检测读出的数据是否出现错误, 若出现错误则将错误计数加 1. 当上述检测完成后, 分别将 0x0 替换为 0x55555555、0xaaaaaaaa 和 0xffffffff 重复上述检测. 最后将错误计数值存入地址 0x40000000 处. 当引导软件进入 main 函数后将错误计数值返回给地面服务程序. 检测 SRAM 的代码如图 4 所示, 进入_checkram 前 %g1~%g4 为 0, %g2 进行错误计数, %5 为 RAM 的大小, %g6 为 RAM 的起始地址.

2) 引导软件自检

引导软件无错误是其正确执行其它功能的前提, 当 RAM 自检正确后, 引导软件将被转移到起始地址为 SADDR 的 RAM 中, 计算其校验和, 若校验和正确则认为引导软件没有错误.

3) 反弹墙保护

反弹墙的目的在于保证程序非法运行时, 能够跳转到指定的异常处理函数. 本软件添加反弹墙的方法是在未使用的区域填入“nop”和跳转到 0 地址的指令, 这些区域包括 ROM、EEPROM、RAM 中未使用的所有区域. 若程序运行中跳入非法地址, 则软件复位.

4) EDAC 保护

EDAC(error detection and correction) 采用海明码, 能够纠正一位错, 检测两位及多位错误, 对于受保护的 RAM 区域, 任何一条指令的读取或数据的读写都将经过 EDAC 检测, 对于一位可纠正错, 错误将在 CPU 内部纠正, 对于两位不可纠正错, 将产生 instruction_access_exception 或 data_access_exception 陷阱.

本引导软件主要对 RAM 进行 EDAC 保护. 软件初始化时, 开启全 RAM 区域 EDAC 保护, 软件运行过程

中, 为了防止比特位的错误累积, 定期对 RAM 中的所有数据进行读写, 若出现多位错误则进入中断服务函数, 中断服务函数将错误发回给地面, 引导软件自动复位.

5) 看门狗

看门狗是防止程序跑飞的有效手段, 在该引导软件中, 引导软件必须定期喂狗, 否则引导软件将复位.

```
_checkram:
        sll     %g4,2,%g7
        add     %g6,%g7,%g6
        st%g1,[%g6]
        nop
        nop
        ld[%g6],%g7
        nop
        nop
        cmp     %g7     ,%g1
        bne     _check_error
        nop
        nop
_continue:
        add     %g4,1,%g4
        nop
        nop
        cmp     %g5,%g4
        bne     _checkram
        nop
        nop
        b_goon
        nop
        nop
_goon:
        add %g3,1,%g3
        cmp%g3,3
        be _save_count
        nop
        nop
        add %g1,0x55555555,%g1
        b     _checkram
        nop
        nop
_check_error:
        add     %g2,1,%g2
        nop
        nop
        b _continue
        nop
        nop
```

图 4 SRAM 检测

3.3 飞行模式

系统自动进入飞行模式后, 对引导标志进行检测, 若导标志不是引导 EEPROM 中的重构程序, 则引导程序将 PROM 中的应用程序复制到起始地址为 SADDR 的 RAM 中, 然后启动应用程序的运行; 若引导标识是引导 EEPROM 中的重构程序, 则引导程序则按照 3.4.2 节中重构程序引导的方式启动重构程序的运行. 飞行模式的流程图如图 5 所示.

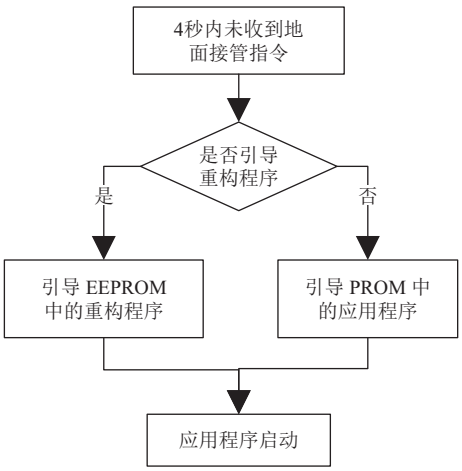

图 5 飞行模式流程图

3.4 测试模式

3.4.1 地面指令

地面指令通过地面服务程序以帧的形式下发到计算机模块, 该计算机模块中的帧分为指令帧、数据帧和应答帧. 指令帧用于计算机模块的控制与测试, 数据帧用于重构程序的上注, 应答帧用于计算机模块状态的回送. 数据的发送往往伴随着封装成帧和帧的重组. 在计算机模块上对帧的处理主要是帧重组和帧解析.

1) 帧重组

帧重组在串口 UART 的中断服务函数中进行. 当收到一个字节的数据时产生 UART 中断, 经中断切换进入中断服务函数. 因数据的发送具有连续性, 进入串口中断服务函数后连续判断 UART 状态寄存器的 dr 位是否为 1, 若此位为 1, 则表示在 UART 的接收保持寄存器中已经有可读的新数据, 串口服务程序读取 UART 数据寄存器中的一字节数据, 并根据帧协议将接收到的数据组装成帧.

2) 帧解析

帧的解析在函数 cmdAnalysis 中进行. 该函数对于一个帧的解析流程如图 6 所示. 若收到的帧是数据帧, 则使用 io_copy 函数将数据 (不包含帧头、帧尾, 帧长度等与帧有关的数据) 转移到首地址为 FRM_BUF_ADDR 的 RAM 中, 数据复制完成后, 计算校验和, 并将其回送至地面服务程序; 若收到的帧是指令帧, 则根据指令码选择相应的操作, 如地面接管、软件上注、重构程序固化、RAM 引导等.

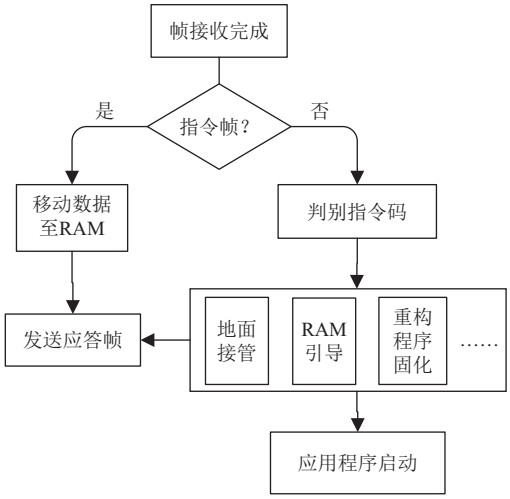

图 6 帧解析流程

3.4.2 重构程序的三冗余架构与引导

该引导软件的一个重要功能是重构程序的在线固化功能. 本引导软件将重构程序分三份固化于 EEPROM 中, 并分别计算校验和, 将校验和发送回地面服务程序.

根据 3.4.1 节对数据帧的论述, 当重构程序存放于 RAM 中, 且地面收到正确的校验和后, 地面服务程序可以发送重构程序固化指令, 开始重构程序的固化. 重构程序的固化使用 FPGA 托管方式进行. 重构程序共固化三份, 首地址分别是 RADDR1、RADDR2、RADDR3, 固化完成后分别计算校验和, 并将校验和返回给地面服务程序, 若应答帧返回的三个校验和与重构程序的校验和相同, 则重构程序固化成功, 此时地面服务程序可以向计算机模块发送重构程序引导指令.

计算机模块收到重构程序引导指令后, 分别将这三份程序按字节取出, 进行三取二操作, 将所得的结果移动到 RAM 中起始地址 SADDR 处. 当程序搬运完毕, 计算 RAM 中该程序的校验和, 若校验和正确, 则启动重构程序的运行, 否则启动原应用程序. 重构程序启动的核心代码如图 7 所示.

4 实验结果

本实验中计算机模块固定在板卡基座上, 实物如图 8 所示, 图 8 左边是计算机模块的正视图, 右边是计算机模块的俯视图. 在试验中使用 EEPROM 代替 PROM, 地面服务程序的界面如图 10 所示.

4.1 引导软件的自动引导

在 tornado 中建立 VxWorks 应用程序, 并添加 printf 打印函数, 设置引导标识, 重启计算机模块, 则可以在串口调试助手中看到如图 9 的结果.

```
for(i=0;i<RLen;i=i+1){
    tmp1=(IO_READ(RADDR1+i*4)&0xff);
    tmp2=(IO_READ(RADDR2+i*4)&0xff);
    tmp3=(IO_READ(RADDR3+i*4)&0xff);
    tmp=twoOfThree(tmp1,tmp2,tmp3);
    IO_WRITEB(SADDR+i,tmp);
}

checksum = calcChecksum(SADDR,RLen);

if(checksum ==0){
    IO_WRITEB(FPGA_STATUS,0x1);
    boot();
}
else{
    appBoot();
}
return;
```

图 7 重构程序引导

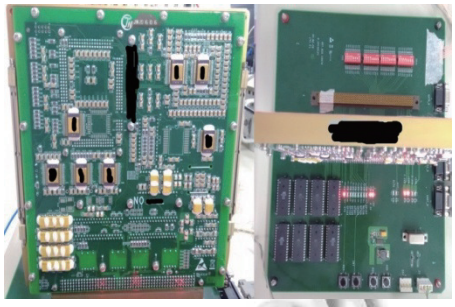

图 8 计算机模块实物图

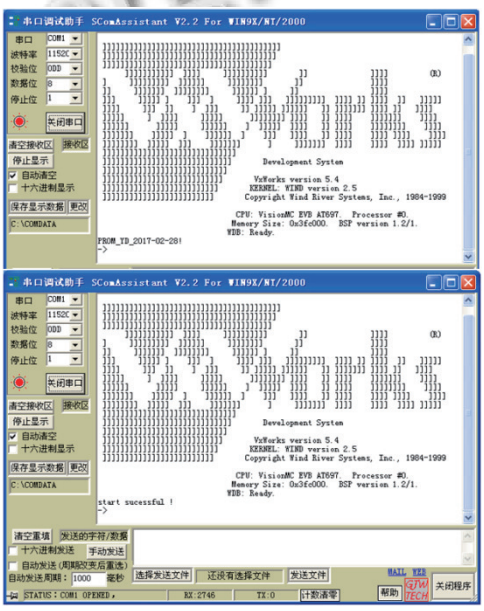

图 9 引导程序自动启动应用程序

图 9 上图为自动引导 PROM 中的应用程序, 下图为自动引导 EEPROM 中的重构程序. 从图中可以看出 VxWorks 已经成功运行, 并成功打印出相应信息, 证明应用程序已经成功运行.

4.2 重构程序的固化与引导

计算机模块重启后发送地面接管指令, 接管成功后, 引导软件进入测试模式, 图 10 显示了地面服务程序发送的一系列地面指令. 从中可以看到程序启动后自检正确, SRAM 无错误. 当发送地面接管指令后成功接管引导软件, 引导程序成功进入测试模式, 这时选取重构程序, 发送软件上注指令, 执行重构程序固化和引导, 从图中可以看出重构程序在计算机、RAM 和 EEPROM 中的校验和完全一样, 都为 0x37b0, 表明引导程序已经成功将重构程序固化完成.

图 10 引导程序进入测试模式

当地面发送重构程序引导指令后, 引导程序引导重构程序的执行. 所得的结果如图 11 所示.

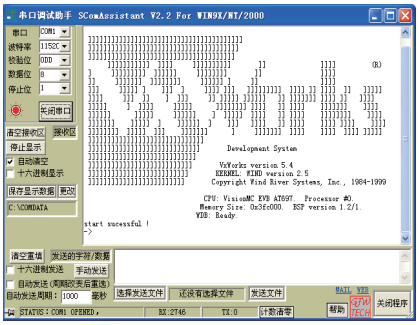

图 11 重构程序引导

4.3 错误检测与恢复

4.3.1 反弹墙测试

在引导软件的 main 函数中添加:

asm(“b 0x40001000”)

运行引导软件, 得到的结果如图 12 所示.

从图 12 中可以看出程序在不停的重启, 说明反弹墙能够正确的运行.

4.3.2 三冗余造错

通过 VDS 将 RADDR1, RADDR2+0xf, RADDR3+0xff 处的值改为 0xFF, 设置引导标识为重构程序引导, 启动引导软件, 可以得到如图 11 所示的结果, 证明三冗余架构具有一定的容错能力, 能够保证应用软件的正常运行.

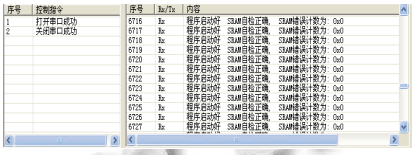

图 12 反弹墙测试

4.3.3 SADDR 处校验和错误

在重构程序放置在 SADDR 后、校验和检测前, 使用 VDS 更改 SADDR 处的值, 运行软件, 此时会检测到校验和错误, 转而引导位于 PROM 中的应用程序, 串口调试助手此时返回的结果如图 9 上图所示.

5 结束语

本文介绍了基于 SPARC 架构的的计算机模块引导软件, 详细论述了该引导软件的架构设计和主要功能, 并论述了为保证其高可靠性采取的技术, 最后将该引导软件在计算机模块上测试, 实验证明, 该引导软件的各项功能正确, 能够将其应用到实际的项目中. 该软件也有不完善之处, 需要改进, 比如为保证高可靠性, 数据的传送速度较慢, 可以通过测试, 在保证高可靠的前提下提高数据传输速度.

参考文献

1 徐福祥, 林宝华. 卫星工程概论. 2 版. 北京: 中国宇航出版社, 2004.

2 AT697F_doc7703.pdf. http://www.atmel.com/Images/doc7703.pdf.

3 陈国林. 星上带容错功能的计算机引导系统的研究和实现[硕士学位论文]. 北京: 中国科学院研究生院 (计算技术研究所), 2005.

4 孙乐益. 嵌入式系统的冗余设计研究. 数字技术与应用, 2011, (10): 123–124.

5 赖鑫. 高可靠 8051 设计与实现及可靠性评估[硕士学位论文]. 长沙: 国防科学技术大学, 2008.

6 柳振华. 三模冗余容错计算机的设计与实现[硕士学位论文]. 西安: 西安电子科技大学, 2010.

7 时晨, 于伦政. 基于 SPARC 结构的 RISC 系统设计技术. 微电子学与计算机, 2002, 19(11): 52–54. [doi: 10.3969/j.issn.1000-7180.2002.11.017]

8 周雪赞, 张宏. 基于 Vxworks 操作系统的 BSP 开发. 工业控制计算机, 2004, 17(2): 33–34.

9 徐惠民. 基于 VxWorks 的嵌入式系统及实验. 北京: 北京邮电大学出版社, 2006.