# 嵌入式系统软件可靠性设计与测试方法①

李金麒[1], 徐建平[2]

[1](上海理工大学 光电信息与计算机工程学院, 上海 200093)
[2](上海工业自动化仪表研究院, 上海 200233)

**摘 要**: 通过设计与初始化 ARM s3c2410 内部的存储器保护系统 MPU 来保护睡眠任务的存储空间不受当前运行任务的非法访问, 有效增强了嵌入式系统软件的可靠性. 并利用 LDRA Testbed 测试套件对项目中的代码进行了静态分析、复杂度分析、图形化分析, 得出了各项测试结果, 准确地评估了所设计出的系统的可靠性, 并给出了优化建议.

**关键词**: 嵌入式系统; 可靠性设计; ARM; 可靠性测试; LDRA testbed

## Design and Test Method for the Reliability of Embedded System Software

LI Jin-Qi[1], XU Jian-Ping[2]

[1](School of Optical-Electrical and Computer Engineering, University of Shanghai For Science and Technology, Shanghai 200093, China)
[2](Shanghai Institute of Process Automation Instrumentation, Shanghai 200233, China )

**Abstract**: Protected sleeping task's cache against illegal calling from currently running task by designing and initiating the MPU in ARM s3c2410. Effectively enhanced the reliability of embedded system software. Then carried on the static analysis, complexity analysis, and graphical analysis for code in the project using the LDRA Testbed, figured out each of the test results, accurately assessed the reliability of the system we designed, and provided suggestions to optimize it.

**Key words**: embedded system; reliability design; ARM; reliability testing; LDRA testbed

## 1 引言

随着信息技术的发展, 嵌入式系统在生产自动化, 信息化中的应用越来越广泛. 然而, 近几年来的统计数据表明, 硬件系统故障率在不断下降, 而软件系统故障率却在不断上升. 软件是绝大多数系统的组成部分, 软件可靠性是嵌入式系统可靠性的构成要素同时也是制约其可靠性的瓶颈. 在当今信息化高度发达的社会, 开发足够可靠的软件并测试和验证其可靠性, 仍然是非常困难的问题, 由软件可靠性问题造成的事故和灾难屡见不鲜, 触目惊心. 所以, 软件可靠性问题已逐渐成为嵌入式系统中亟需解决的问题.

软件可靠性(Software Reliability)是最重要的软件质量特性之一, 是系统可依赖性(System Dependability)的关键因素. IEEE 计算机学会于 1983 年做出正式定义: 在规定的条件下和规定的时间内, 软件不引起系统失效的概率. 该概率是系统输入和系统使用的函数, 也是软件中固有错误的函数简单的说, 就是在规定的条件下和规定的时间内, 软件执行规定功能的能力.

## 2 基于MPU的嵌入式系统软件可靠性设计

一些嵌入式系统使用多任务的操作或控制系统, 在这种高复杂的系统中, 正在运行的任务会有破坏其他任务的可能性, 在设计系统时, 需要防止系统资源和其他任务受非法访问, 以提高嵌入式系统软件的可靠性.

嵌入式 ARM 处理器配备了有效保护系统资源的硬件: MPU(Memory Protection Unit),通过对软件定义的区域提供硬件保护来提高系统的可靠性.

在受保护系统中, 主要有两类资源需要监视: 存

① 收稿时间:2012-06-03;收到修改稿时间:2012-08-26

储器系统和外围设备. ARM 的外围设备通常都映射到存储器中, MPU 就是通过使用这样的方法来保护这两类资源. MPU 是一个内存保护的可选组件, 可以用来将内存划分为多个区域, 并为各个区域设置独立的访问权限. 本文通过嵌入式开发项目中的存储器保护系统的设计来介绍嵌入式系统软件的可靠性设计.

### 2.1 系统概况

本系统的硬件特征如下:

①Samsung s3c2410 带 MPU 的 ARM 核

②512k 的物理存储器.

③若干存储器映射的外设, 位于 0x1000000~0x1400000 的 4M 字节空间中.

本系统的软件模块如下:

①系统软件小于 128K, 包括内存读写操作函数, 端口操作函数, 进程控制函数等.

②有一个小于 128k 的共享程序, 包括通用库和用户任务间传递消息的数据空间.

③有 3 个在系统中控制独立功能的用户任务, 这些任务每个小于 64k, 其中一个任务在运行时, 不能被另外两个任务访问.

### 2.2 使用存储器映射分配区域

编写代码时, 将各个软件模块放在分配给它们的区域内. 系统模块的访问权限是系统级的, 共享程序的空间可被整个系统访问, 任务程序区域包含用户级任务. 本系统的 3 个用户任务映射在区域 3, 受保护的系统和外围设备的访问级别为系统级, 分别映射在区域 1 和区域 4, 如图 1 所示.

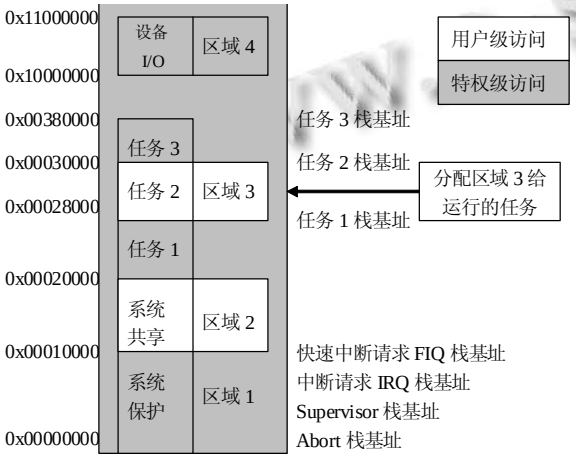


图 1 区域分配和受保护系统的存储器映射

区域 1 是一个背景区域, 覆盖整个可寻址的存储器空间. 它的主要功能是限制对受保护的系统区域访问, 即 0x0-0x10000 之间的 64k 空间.

区域 2 位于从 0x10000 开始的 64k 空间中, 映射在共享系统代码的共享存储器上, 控制对共享系统资源的访问.

区域 3 用来控制运行任务的存储器空间和属性. 当控制权从一个任务传给另一个任务时, 即任务切换, 操作系统就重新定义该区域, 使其覆盖运行任务的存储器空间.

区域 4 是存储器映射的外围系统空间. 该区域用来建立没有 cache 和没有写缓冲器的空间. 这样在对控制寄存器和 I/O 设备操作时可以避免使用写缓冲导致的时间或优先级问题.

### 2.3 配置系统的 MPU

为了初始化 MPU,cache 和写缓冲器, 控制系统必须定义在操作目标平台时所需的保护区域.

控制系统通过配置寄存器 c2 和 c3 来设置区域的 cache 和写缓冲器的属性, 寄存器 c5 控制区域的访问权限, 在寄存器 c6 里有 8 个或 16 个次寄存器, 用来定义每个区域的大小和位置. 图 2 说明 CP15:c6:c0-CP15:c6:c7 的 8 个次寄存器的位域和格式.

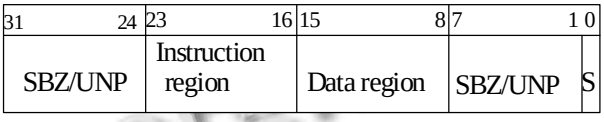


图 2 设置区域的大小和位置的 CP15: c6 寄存器格式

其中, SBZ/UNP 表示置 0 或无法预知, Instruction region 用来设置指令域的大小, 一般设为 b00. Data region 作为指定数据域大小, 若 MPU 存在则置为 0x10, 否则置 0. S 位指定 MPU 的类型.

### 2.4 设置访问权限

ARM 有 2 组可用的访问权限机制: 标准组和扩展组. 标准组有 4 级访问权限, 扩展组 AP(access permission)增加了额外的 2 级权限,通过写 CP15:c5 的次寄存器来分配区域的访问权限. 次寄存器 CP15:c5:c0:0 和 CP15:c5:c0:1 配置标准的 AP, 次寄存器 CP15:c5:c0:2 和 CP15:c5:c0:3 配置扩展的 AP, 如图 3 所示.

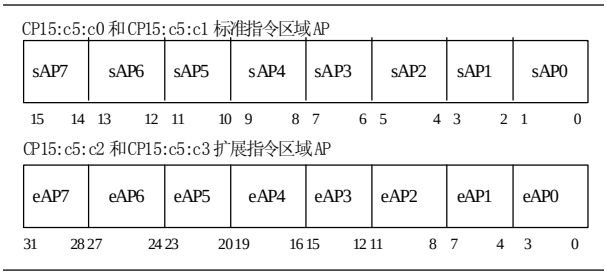


图 3　CP15 寄存器 5 访问权限寄存器格式

本系统通过对 CP15:c5 寄存器进行读-修改-写操作来设置指定区域的 AP. MCR/MRC 指令决定指令和数据访问权限位是否可访问，并决定是否对标准或扩展寄存器进行读写:

```
MRC p15, 0, rd, c5, c0, 2;   //读取数据访问权限位
MRC p15, 0, rd, c5, c0, 3;   //读取指令存取权限位
MCR p15, 0, rd, c5, c0, 2;   //写数据访问权限位
MCR p15, 0, rd, c5, c0, 3;   //写指令存取权限位
```

指令访问权限 IApn 和数据存取权限 DApn 确定内存各个区域的访问权限，访问权限编码如表 1:

表 1　IApn/DApn 访问权限编码

| 扩展类型 | | 标准类型 | |
|---|---|---|---|
| I/DApn[3:0] | 访问权限 | I/DApn[1:0] | 访问权限 |
| 0000 | 无权限 | 00 | 无权限 |
| 0001 | 无权限 | 01 | 无权限 |
| 0010 | 只读 | 10 | 只读 |
| 0011 | 可读可写 | 11 | 可读可写 |
| 0100 | UNP | | |
| 0101 | 无权限 | | |
| 0110 | 只读 | | |
| 0111 | UNP | | |
| 1xxx | UNP | | |

### 2.5 系统初始化

在初始化 MPU 前，需要配置好协处理 CP15 的所有相关的寄存器，包括建立至少一个内存区域，并将高速缓冲存储器 cache 清空.

接着创建一个称为 Area 的结构，该结构的成员保存系统操作中使用的区域的属性，称此结构为区域控制块 ACB(Area Control Block).

```
typedef struct{
unsigned int Number;    /*MPU 区域号*/
unsigned int Type;      /*访问权限类型*/
unsigned int Baseaddress; /*起始地址*/
unsigned int Size;      /*区域大小*/
unsigned int IAP; /*指令寄存器访问权限*/
unsigned int DAP; /*数据寄存器访问权限*/
unsigned int CB;    /*写缓冲寄存器配置*/
    }Area;
```

输入参数为一个指向区域 ACB 的指针，内部使用 Area 的成员项作为初始化过程的数据输入.

```
void configArea(Area *area)
{
/*Step1: 使用 cp15:c6 来定义指令和数据区的大小和位置*/
    areaSet(area->Number, area->Baseaddress,
      area->size, R_DISABLE);
/*Step2: 使用 cp15:c5 来设置每个区域的访问权限*/
    if(area->type = = STANDARD) //标准类型
    {
    areaSetSIAP(area->Number, area->IAP);
    areaSetSDAP(area->Number, area->DAP);
    }
    else if(area->Type = = EXTENDED) //扩展类型
    {
    areaSetEIAP(area->Number, area->IAP);
    areaSetEDAP(area->Number, area->DAP);
    }
    /* Step3: 使用 CP15:c2 和 CP15:c3 来设置每个区域的 cache 和写缓冲器属性*/
    areaSetCB(area->Number,area->CB);
/* Step4: 使用 CP15:c6 和 CP15:c1 来使能 cache、MPU 和写缓冲器*/
    areaSet(area->Number, area->Baseaddress,
      area->size, R_ENABLE);
}
```

## 3　基于LDRA Testbed测试套件的嵌入式系统软件可靠性测试

LDRA 软件测试工具采用国际上使用的基于软件度量的质量模型，支持多种软件编程规则；使用代码覆盖率检验软件测试效率；主要工作有：基本静态分

析、复杂度分析、软件度量、数据的图形化显示、文字包含.

### 3.1 复杂度分析

在硬件的可靠性设计中, 有一条基本原则“简单就是可靠”, 这个原则同样也适合软件. 软件的复杂度是影响软件可靠性的一项重要指标.

当两个控制流程交叉时会产生节点, 在 c 语言中易引起节点的程序结构有 for、while 循环以及 switch 分支结构. 如果节点太多, 将会增加程序的复杂性, 图 4 为从所设计的软件 s3c2410_ts.c 中提取的读内存模块 s3c2410_ts_read 的控制流图, 控制流图研究的是函数最基本的控制流结构, 程序中的源码块表示为节点, 各节点间的跳转用连线表示. 菱形节点表示该节点包含的代码存在违反编程规则的情况.

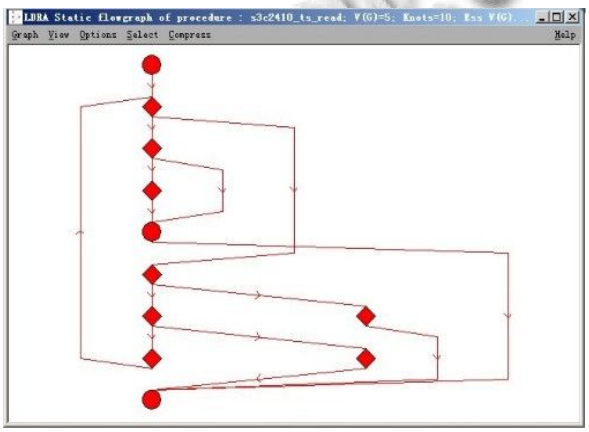


图 4 读内存模块 s3c2410_ts_read 的控制流图

复杂度度量是测量在一个软件模块中的分支数目, 在所有的开发周期中都要使用. 圈复杂度度量以软件的结构流程图为基础, 用来衡量一个模块判定结构的复杂程度. 经验证明圈复杂度越大, 程序越复杂, 可靠性越差. 图 5 为系统软件 s3c2410_ts.c 各个子模块的圈复杂度.

### 3.2 LCSAJ 密度和代码可达性

LCSAJ(Linear Code Sequence and Jump)线性代码序列及跳转, 是一个可执行代码的线性序列, 作为软件维护性度量指标, 可以识别不可测试或不合理的代码, 如果从程序开始没有任何路径到达指定的代码行, 那么代码将被标为不可达(或死代码). 如果一行代码被修改,LCSAJ 密度将指出多少测试路径会受到此改动的影响, 若 LCSAJ 密度过高, 则需要的回归测试的次数相应增加. 图 6 为系统软件 s3c2410_ts.c 的 LCSAJ 测试报告总述, LCSAJ 总数为 61, 可达 LCSAJ 数为 60, 不可达 LCSAJ 数为 1, 最大 LCSAJ 密度为 6, 不可达代码为 2, 不可达分支为 2.

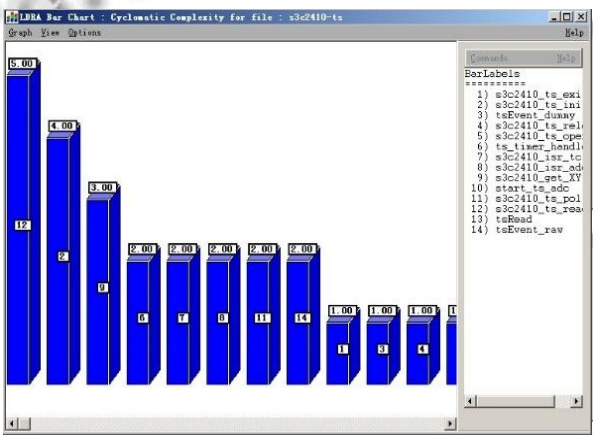


图 5 s3c2410_ts.c 各个子模块的圈复杂度

### 3.3 调用关系图

当前形势下, 嵌入式系统软件愈发复杂化, 需要将软件模块化以便于软件的开发、测试及维护. 调用关系图通过显示模块(函数)间的调用关系, 表示出被测软件的整体结构, 这种调用关系图可以是针对单个文件的, 也可以是针对整个工程的. 在 LDRA Testbed 中选择 Static Callgraph 即可以生成程序 s3c2410_ts.c 中的函数调用关系图, 如图 7. 图中深色的节点为程序的内部函数; 浅色的节点为程序的外部函数.

Summary Information

| TOTAL LCSAJS | REACHABLE LCSAJS | UNREACHABLE LCSAJS | MAX. LCSAJ DENSITY | UNREACHABLE LINES | UNREACHABLE BRANCHES |
|---|---|---|---|---|---|
| 61 | 60 | 1 | 6 | 2 | 2 |

图 6 s3c2410_ts.c 的 LCSAJ 测试报告总述

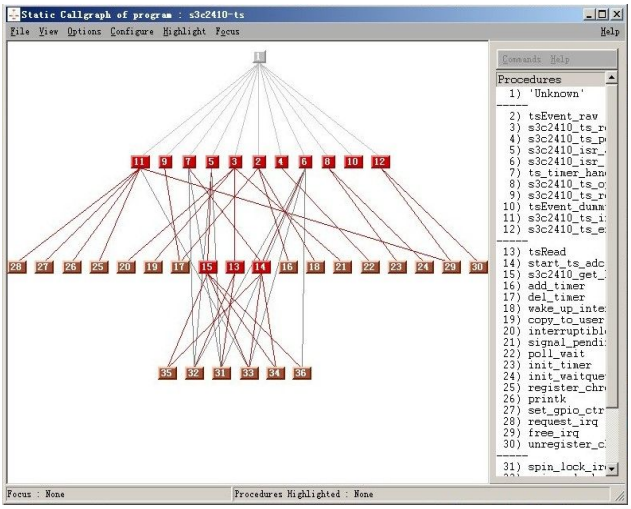


图 7 s3c2410_ts.c 中的函数调用关系图

系统调用关系图将整个程序内部的调用关系直观地展现出来，使得软件的层次更加清晰，同时，在软件的各个测试阶段，系统调用关系图还可以显示软件模块划分的详情，这对于提高嵌入式系统软件可靠性具有重要帮助.

### 3.4 测试质量报告

LDRA Testbed 提供了较为详细的质量分析报告，可以直接反应被分析源代码的质量，下图是 s3c2410_ts.c 的质量分析报告，从报告中可看到各模块的名称、质量分析结果、所属源文件和违反规则情况，也可通过链接直接定位违反规则的代码进行分析. 其中,s3c2410_ts_read,start_ts_adc,s3c2410_ts_release 这 3 个函数模块的失效密度较大，分别为 45, 40, 40, 表明这 3 个模块容易出现问题，建议对其进行优化以提高可靠性.

| Quality Result | Procedure | Source File | Unique Standards Failure Ratio (%) | Failure Density (Viols/K.Line %) |
|---|---|---|---|---|
| FAIL | Global Program | | | |
| FAIL | tsEvent_raw | s3c2410-ts.c | 4 | 15 |
| FAIL | tsRead | s3c2410-ts.c | 5 | 38 |
| FAIL | s3c2410_ts_read | s3c2410-ts.c | 10 | 45 |
| Conditional Pass | s3c2410_ts_poll | s3c2410-ts.c | 1 | 9 |
| FAIL | start_ts_adc | s3c2410-ts.c | 3 | 40 |
| FAIL | s3c2410_get_XY | s3c2410-ts.c | 4 | 25 |
| FAIL | s3c2410_isr_adc | s3c2410-ts.c | 2 | 18 |
| FAIL | s3c2410_isr_tc | s3c2410-ts.c | 2 | 10 |
| Conditional Pass | ts_timer_handler | s3c2410-ts.c | 1 | 8 |
| FAIL | s3c2410_ts_open | s3c2410-ts.c | 3 | 30 |
| FAIL | s3c2410_ts_release | s3c2410-ts.c | 2 | 40 |
| Pass | tsEvent_dummy | s3c2410-ts.c | 0 | 0 |
| FAIL | s3c2410_ts_init | s3c2410-ts.c | 7 | 30 |
| Pass | s3c2410_ts_exit | s3c2410-ts.c | 0 | 0 |

图 8 s3c2410_ts.c 的质量分析报

## 4 结语

ARM 中的 MPU 使用区域作为系统保护的主要概念，用来增强嵌入式系统软件的可靠性，本文详细介绍了针对 samsung 公司的 s3c2410 芯片的 MPU 设计与初始化方式,设计了一个多任务环境系统，系统有 3 个任务，有效保护了每个任务不受其他 2 个任务的非法访问.

接着运用 LDRA Testbed 测试套件对所设计的项目中烧写到 ARM 处理器中的嵌入式软件 s3c2410_ts.c 进行了各项测试分析并生成了测试质量报告，有效评估了该软件的可靠性. 然而，在对检测到的不符合可靠性标准的模块，尚未有一个有效解决方案，今后的工作将着重研究这方面的内容.

## 参考文献

1 孙志安,裴晓黎,宋昕,戴忠健.软件可靠性工程.北京:北京航空航天大学出版社,2009,176–191.
2 蔡建平.嵌入式软件测试实用技术.北京:清华大学出版社,2010:230–253.
3 Sloss AN, Symes D, Wright C. ARM 嵌入式系统开发.北京:北京航空航天大学出版社,2005.445–471.
4 LDRA Testbed 中文使用指南.
5 石磊.LDRA TestBed 在弹载软件测试中的应用.软件导刊,2008,5:11–13.
6 邹会荣.基于 LDRA Testbed 的飞机中央维护系统覆盖测试.航空计算技术,2010,5:91–94.