嵌入式系统代码静态分析与测试方法探讨

Study on Code Static Analysis and Code Test Technology of Embedded System

何天添[1]　凌志浩[1,2]

（华东理工大学信息学院[1]，上海　200237；化工过程先进控制与优化技术教育部重点实验室[2]，上海　200237）

摘　要：随着嵌入式系统硬件可靠性水平的不断提高和嵌入式软件的日益复杂化，软件的可靠性在嵌入式系统整体可靠性中所占的比重也越来越大。在探讨了代码静态分析技术中关键步骤的基础上，制定了编码标准，并且选择了合适的静态分析工具；同时，介绍了自行开发的代码测试框架和测试用例设计方法，并将其运用到某智能仪表软件的测试中，得到了良好的测试效果。

关键词：嵌入式系统　软件可靠性　静态分析　代码测试　XML 技术

中图分类号：TP206 + .1　　**文献标志码**：A

Abstract: Along with the situation that the reliability of the hardware of embedded system is continuously increasing and the embedded software is becoming more complicated, the reliability of software plays more important role in integrated reliability of the embedded system. The critical procedure of code static analysis technology is investigated and the encoding standard is drew up, and the static analysis tool is selected. In addition, the self-developed code test framework and design method of the test example are introduced. These have been applied in software test for certain intelligent instrument with excellent test effects.

Keywords: Embedded system　Software reliability　Static analysis　Code test　XML technology

0　引言

以通信和计算机技术为基础的嵌入式系统正越来越多地被应用于工业过程控制领域中，随着工业生产环境的复杂恶劣和人类“安全生产”意识的提高，对嵌入式控制设备的可靠性和安全性提出了更高的要求。同时，考虑到现今嵌入式系统中越来越多的控制功能均由软件实现的情况，系统的高可靠性和安全性实现应着重考虑软件部分。

通常，软件开发遵循 V 模型，主要包含设计和验证两大环节。测试作为验证环节不可缺少的部分之一，其开销较大，特别是对于嵌入式系统测试，其本身就具有很高的难度。虽然 IEC61508 功能安全标准给出了验证环节应采用的技术和措施[1]，但具体的实现方法和实施细节仍需我们去探索。编码阶段实现由设计到代码的转变，是 V 模型中的一个重要阶段。由于编码直接关系软件运行的可靠性，因此，本文主要讨论代码静态分析方法和模块测试技术。

代码静态分析是指在代码复审阶段，依靠代码静态分析工具，无需执行程序即可发现可能包含错误或漏洞的代码。它可以使代码中的缺陷数降低 6 倍[2]。代码静态分析除了需选择合适的静态分析工具外，制定有效、可实施和可验证的编码标准也是关键之处。模块测试则需要运行被测代码，搭建测试环境和设计合适、有效的测试用例。

1　代码静态分析技术

嵌入式软件设计一般均采用模块化方法，静态分析所针对的对象就是各个模块，通常在静态分析之后即对模块进行测试。代码静态分析的一般过程如图 1 所示，其中比较重要的步骤为制定编码标准、选择合适的静态分析工具及结果检查等。

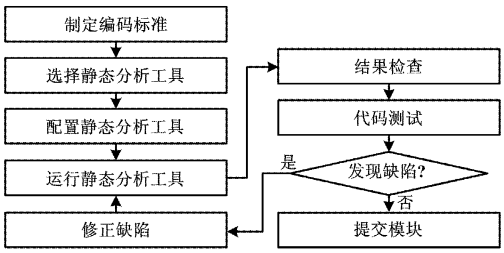

图 1　代码静态分析的一般过程

Fig. 1　General process of code static analysis

1.1　编码标准制定

代码静态分析的第一步需要制定编码标准。虽然编码标准并无精确定义，但 IEC61508 认为编码标准应

国家“863”基金资助项目（编号：2009AA04Z144）；
浙江省科技计划基金资助项目（编号：2008C16016）；
上海市重点学科（B504）基金资助项目。

第一作者何天添，男，1986 年生，现为华东理工大学控制理论与工程专业在读硕士研究生；主要从事功能安全技术方面的研究。

规定较好的编程习惯、限制非安全语言特性（如未定义的语言特性、非结构化设计等）和规定源代码文档规程。

由于语言本身可能包含许多未定义或实现定义行为，所以制定编码标准是一个相当繁琐的过程。一般的做法是参考现有的标准。目前较为流行的是MISRA C。该标准是英国 MISRA 协会于 1998 年发布的、针对汽车工业软件安全性的 C 语言编程规范[3]。在 2004 年的新版本中，共有强制规则 121 条，推荐规则 20 条。尽管该标准中采用其自定义的术语，如“基本类型”、“复杂表达式”等，但仍有某些规则较为模糊，同时 2004 新版中规则的误判率也并未下降[4]。但 MISRA C 仍在汽车工业中得到普及且已扩展到嵌入式开发的其他方向。此外，许多工具商也推出了包含 MISRA C 检查的代码静态分析工具，如 TestBed、QAC 和 PC-Lint 等。因此，可在 MISRA C 的基础上制定新的编码标准，并根据项目开发的语言、平台和编译环境的特点，对规则给出合理的解释（如某规则是如何帮助提高安全性、示例代码和例外情况等）。

另外，可供参考的编码标准还有 CERT C[5]。这是 2008 年由美国国防部和卡内基梅隆大学软件工程研究院共同推出的 C 安全编码标准，旨在通过其安全编码规则和建议，减少不安全的编码实践和不确定的行为，增强程序的“健壮性”。LDRA 公司推出的 TBsecure 组件声称已实现 CERT C 安全编码检查。国内也有一个相关的标准，即由航天软件评测中心制定的《GJB5369－2005 航天型号 C 语言安全子集》。

1.2　静态分析工具选择

选择合适的静态分析工具是代码静态分析过程中的一个重要环节。至今为止，市面上已有数十种针对不同语言的静态分析工具，这些工具提供的功能大致包括代码规则检查、静态分析检测和软件质量度量等。其中，代码规则检查主要侧重于检查代码是否符合当前比较流行的编码标准，如 ISO、MISRA C 、EC ++ 等。静态分析检测主要是通过建立源代码的抽象表示，如 AST（抽象语法树），进而分析引擎查出可疑代码。软件质量度量则侧重于从宏观方面考察源文件的质量，采用的评价准则主要是当前比较流行的 McCabe 复杂度、Halstead 程序度量、注释率和规模等。

针对如此多的代码静态分析工具，通常可从以下 4 个方面加以考查选择：①适用语言、平台和编译环境；②代码规则检查、软件质量度量、静态分析等功能；③是否提供分析报告，如 XML、HTML 格式；④是否简单易用。

需要注意的是，尽管不同工具提供相同的功能，但其功能实现的深度一般有所差别。如 Klockwork、QAC 和 PC-Lint 均提供静态检测功能，但它们检测的深度是不一样的。同时，对于隐藏较深的 Bug 给出的结果可能也不一致，这主要取决于其采用的分析引擎。代码静态分析工具价格普遍较高，相对而言，PC-Lint 的价格较低，并且功能较为齐全，可方便地嵌入到主流编译环境中。如要将 PC-Lint 嵌入到 Visual C ++ 6.0 环境，只需在其“外部工具”选项中新建一个名为“PC-Lint”的工具，将工具的路径指定为 lint-nt. exe 文件所在目录，同时添加参数 “-u”、std. lnt 和 env-vc6. lnt 文件路径以及待检查文件路径即可。在 Codewarrior 的专业版中，由于其本身就提供 PC-Lint 嵌入功能，因而更为方便，只需在“PC-Lint main settings”对话框中选择 lint-nt. exe 文件所在路径，同时选择编译器类型，即可将 PC-Lint 集成在 Codewarrior 中。

1.3　结果检查

由于编码标准中部分规则的模糊性以及工具本身能力的局限性，不可避免地会存在代码静态分析工具所不能检查的规则，如 PC-Lint v9.0 不能支持 2004 版本 MISRA C 中的 27 条，检测率约为 80%。因此，为了确保所有的规则均被覆盖到，需要产生一个符合性矩阵，矩阵中列出每条规则并说明它是如何被检查的。表 1 为在 Codewarrior 平台下结合 PC-Lint 工具生成的符合性矩阵示例。

表 1　符合性矩阵示例

Tab. 1　Example compliance matrix

规则号	强制/建议	检测工具			备注
		Codewarrior	PC-Lint	人工	
9.1	强制		是		
13.1	强制	是			
13.5	强制			是	
13.2	建议		是		

对于未能符合的规则，如与微处理器硬件接口的定义将不可避免地使用联合，应给出详细的背离过程说明（包括被背离的规则、由背离产生的潜在后果以及对背离的正确性声明等）[3]。

2　代码测试技术

2.1　嵌入式模块代码测试框架

由于系统内部资源有限且硬件相关度高等特点，加大了嵌入式软件测试的难度，尤其是对于规模较小或中等的嵌入式系统。目前，可供参考的嵌入式软件测试工具大多面向规模较大的系统，导致其在运行时需占用较多的资源，如 C ++ Test 提供的针对嵌入式测

试的工具生成的测试代码，其需要占用至少 256 kB 的内存空间，这对一般的嵌入式系统来说要求过高。因此，针对规模一般的系统，我们可参考 Unity 单元测试框架[6]，自行搭建一个嵌入式模块代码测试框架。该框架可自动运行测试用例、给出测试结果并生成测试报告，且占用的存储空间不超过 25 kB，减少了对被测系统的影响。

嵌入式模块代码测试框架如图 2 所示，主要包括驱动、判定测试输出和测试报告生成三个部分。根据待测模块设计测试用例后，驱动模块将自动导入针对该模块的所有测试用例。运行时，判定测试输出模块将自行统计测试结果信息，最后生成测试报告。

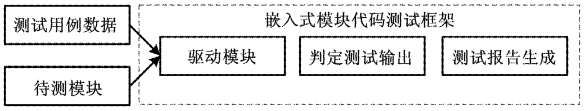

图 2　嵌入式模块代码测试框架

Fig. 2　Test framework of embedded software module

2.1.1　驱动模块

驱动模块主要包括对框架本身的初始化，在调用被测模块的同时送入用例数据，以及对异常情况的处理。如程序有时会因错误条件而进入死循环，这里采取的办法是当测试失败时立即跳出被测程序，进入预定的异常处理程序。异常处理主要由宏 TEST_PROTECT 和 TEST_ABORT 实现。

2.1.2　判定测试输出

由于输出变量可以为多种数据类型，如整型、浮点型、布尔型、数组、指针和字符串等，因此，针对不同的数据类型，我们自定义了一个断言集，以方便用户使用。该断言集包括 TEST_ASSERT_TRUE_MESSAGE (condition, message)、TEST_ASSERT_EQUAL_INT (expected, actual)、TEST_ASSERT_FLOAT_WITHIN (delta, expected, actual)等。

2.1.3　测试报告生成

除了可在下位机同步显示测试结果信息(Codewarrior 的 TERMINAL 组件)外，测试框架同时可提供测试报告的自动生成功能。该功能是在上位机实现的。报告所包含的信息主要是每个测试用例的执行情况，一旦失效，则会给出失效发生时所在的被测文件、代码行号和失效原因等信息，以及测试总体情况、所有用例通过或发生多少失效等。考虑到今后可能需将测试结果信息导入数据库以实现对测试数据的管理，报告的格式采用了目前较为流行的 XML 文档格式。

测试框架主要具有以下优点。

①与被测代码独立，直接调用被测模块，只关心其接口形式。

②可移植性好，除了 SCI 异步通信部分与微处理器相关，其余代码均采用 ANSI C 编写。因此，只需修改 SCI 通信部分程序即可移植到其他开发平台上。

③占用资源尽可能少，通过对需占用较多系统资源的库函数(如 PRINTF、ITOA 等)进行改写，有效降低了对系统内部资源的占用。

2.2　测试用例设计方法

测试流程图如图 3 所示。

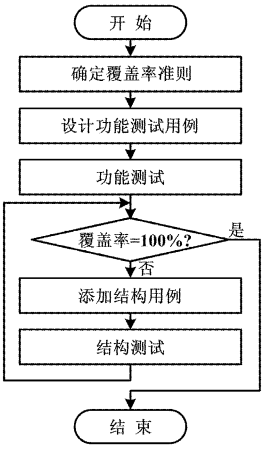

图 3　模块代码测试流程

Fig. 3　Flowchart of module test

设计合理、有效的测试用例至关重要，它直接关系到测试结果的好坏。由于纯粹的黑盒测试难以实现对算法内部具体实现的考查，因此，在模块测试环节中一般以结构测试为主。

结构测试则主要针对程序流程，以满足一定的覆盖率准则来设计测试用例，其主要目标是实现对程序路径的 100% 覆盖。根据覆盖的精确程度，可分为入口覆盖(entry point coverage)、语句覆盖(statement coverage)、判定覆盖(decision coverage)和修正条件/判定覆盖 MC/DC (modified condition/decision coverage)等[7]。

在 IEC61508 2010 新版中，对于安全完整性等级为 SIL2 的软件，其结构测试的要求为 HR(强烈推荐)的有 100% 入口覆盖和 100% 语句覆盖。判定覆盖和修正条件/判定覆盖分别只在 SIL3 和 SIL4 等级下为强烈推荐。因此，我们主要以 100% 语句覆盖作为用例设计的依据。同时，考虑到嵌入式软件路径较多，直接进行结构测试用例的设计较为复杂，我们采取的策略是针对需求说明书先进行功能测试，再根据统计到

的覆盖率信息补充用例,以实现100%语句覆盖。

3　代码静态分析和测试的实现

应用上述测试技术,在Codewarrior平台下结合PC-Lint和嵌入式模块代码测试框架,设计测试用例,进行代码测试。下面结合对某智能仪表中计算数字电流值模块的测试,说明测试过程的实现。待测程序模块的源代码如下。

```
#include "CAL_CURRENT.h"
                                    //计算数字电流子程序
                //输入 PERCENT0,Device_status;输出 X2_0
float32_t CAL_CURRENT(float32_t PERCENT0)
{
  float32_t X2_0;
  if (Device_status.CurrentFixed = =1u)
  {
    X2_0 = 16.0f;                   //电流固定为16 mA
  }
  else if(Device_status.Malfunction = =1u)
  {
    if(Device_status.AlarmSel = =0u)
    {
    X2_0 = 22.0f;              //高报警,报警值为22 mA
    }
    else
    {
    X2_0 = 3.8f;               //低报警,报警值为3.8 mA
    }
  }
  else
  {
  X2_0 = 4.0 + (16.0f * PERCENT0);
                                    //计算数字电流
  if (X2_0 <3.9f)
    {
  X2_0 = 3.9f;               //电流过小,则固定为3.9 mA
    }
    else if (X2_0 > 20.8f)
    {
    X2_0 = 20.8f;          //电流过大,则固定为20.8 mA
    }
else {}
  }
  return X2_0;
}
```

待测程序实现的功能为根据百分量程值计算数字电流值,当电流值高于20.8 mA或低于3.9 mA时,则分别将其限定在20.8 mA或3.9 mA;根据待测程序中的Device_status中各个标志位的信息,作出相应动作,包括固定输出电流值、输出电流高报警或输出电流低报警。

3.1　代码静态分析的实现

代码编写完成后,选择Codewarrior“Target Setting”中的linker为PC-Lint linker,同时选择“PC-Lint Options”中的MISRA选项,编译链接即可实现对代码的MISRA C编码标准检验。经调试,上述源代码已通过检验。

需要注意的是,由于系统自带的库文件(如外部控制寄存器定义和启动文件)一般是混编的,既包含汇编代码,同时也包含非ANSI C的预定义标志符,它们是不被PC-Lint识别的,因此,会导致这两个文件中报错较多。通常的做法是应用PC-Lint提供的错误隐藏命令(如常用的-e#、! e#、-efunc、-elib等)将错误隐藏即可。

3.2　代码动态测试的实现

如图4所示为程序流程图设计测试用例。

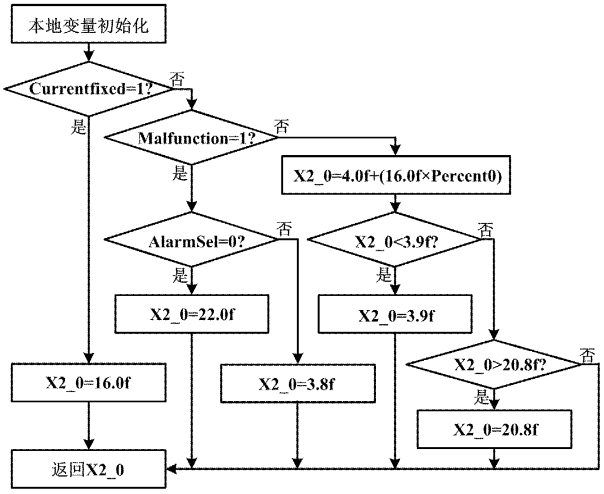

图4　示例程序的流程图

Fig.4　Flowchart of example code

通过PC-Lint的检查后,即需进行模块测试。具体步骤如下。

① 将待测函数导入代码测试框架,同时配置测试环境,使其与开发环境相同。由于该模块没有调用其他函数,因此不需要建立桩模块。

② 修改驱动模块中的驱动函数,与待测函数连接。

③ 配置上位机通信参数,建立与下位机的通信。

运行测试后,代码测试框架可自动依次执行每个测试用例,同时检查实际输出结果是否与预期输出结果一致,测试完成后可生成XML格式的测试报告。

在本示例程序中共包含6条路径。因此,要想实

现100%语句覆盖至少需要设计6个测试用例,每个用例必须包含预期输入和预期输出,并分别覆盖一条路径,具体参见表2。

表2　示例程序的测试用例数据

Tab.2　Test cases for example code

序号	预期输入	预期输出
CAL_CURRENT_TestCase1	Percent0 = 0.50	X2_0 = 12.0
CAL_CURRENT_TestCase2	Device_status. CurrentFixed = 1	X2_0 = 16.0
CAL_CURRENT_TestCase3	Device_status. Malfunction = 1, AlarmSel = 1	X2_0 = 3.80
CAL_CURRENT_TestCase4	Device_status. Malfunction = 1, AlarmSel = 0	X2_0 = 22.0
CAL_CURRENT_TestCase5	Percent0 = −0.007 < −0.00625	X2_0 = 3.9
CAL_CURRENT_TestCase6	Percent0 = 1.1 > 1.05	X2_0 = 20.8

在测试报告中可看到每个用例的执行情况和该次测试的总体信息。关于测试覆盖率的信息,将由Codewarrior调试环境自带的Coverage组件统计,如果语句覆盖未达到100%,则需说明对该模块的逻辑结构未实现完全覆盖。

4　结束语

在当今对嵌入式软件可靠性要求越来越高的背景下,参考软件开发V模型进行嵌入式软件的设计和验证是提高软件可靠性的一个可行思路。本文主要从软件开发V模型的编码阶段出发,探讨嵌入式软件代码静态分析和动态测试技术的实现,着重讨论了编码标准的制定、静态分析工具的选择和测试用例设计等问题,并针对智能仪表等片上资源较为紧缺的嵌入式软件开发了一个代码测试框架。该框架以较少的资源占用实现了基本的测试功能。

参考文献

[1] IEC. IEC 61508 −3 ed.2 Functional safety of electrical/electronic/programmable electronic safety-related systems part 3: software requirements[S]. 2010: 47 −59.

[2] Xiao S, Pham C. Performing high efficiency source code static analysis with intelligent extensions [C]//Proceedings of the 11th Asia-Pacific Software Engineering Conference, Busan, Korea, 2004: 346 −355.

[3] MISRA Limited. MISRA C: 2004 guidelines for the use of the C language in critical systems [CP/OL]. [2005 −02 −05]. http://www.misrac.com.

[4] Hatton L. Language subsetting in an industrial context: a comparison of MISRA C 1998 and MISRA C 2004 [J]. Information and Software Technology, 2007, 49(5): 475 −482.

[5] Carnegie Mellon University. CERT C programming language secure coding standard [CP/OL]. [2007 −12 −28]. http://www.cert.org/secure-coding.

[6] Williams G. Unity compact test framework for C[CP/OL]. [2009 −12 −07]. http://sourceforge.net/apps/trac/unity/wiki.

[7] Baresi L, Pezze M. An introduction to software testing[J]. Electronic Notes in Theoretical Computer Science, 2006, 148(1): 89 −111.

(上接第25页)

展,高性能、低价格的微波物位测量仪器的推出则使其在石油、化工、冶金等众多领域得到广泛的应用,拥有广阔的发展前景[8]。

参考文献

[1] 李竞武. 物位测量新技术及我国的物位仪表行业概况[J]. 中国仪器仪表, 2007(9): 5.

[2] 刘贵忠, 邸双亮. 小波分析及其应用[M]. 西安: 西安电子科技大学出版社, 1992: 124 −130.

[3] 胡翔, 王东进. 一种提高LFMCW雷达调频线性度的新思路[J]. 中国科学技术大学学报: 自然科学版, 2001, 31(1): 63 −67.

[4] 张贤达, 保铮. 非平稳信号分析与处理[M]. 北京: 国防工业出版社, 1998: 32 −35.

[5] 陈淑珍, 谢秋洪, 肖柏勋. 时间尺度域匹配滤波器的设计与实现[J]. 武汉大学学报: 理学版, 2005, 51(1): 91 −93.

[6] 顾明超, 王雷, 赵国庆. 基于小波分析的雷达信号调制方式识别[J]. 舰船电子对抗, 2009, 32(6): 83 −85.

[7] 张振宇, 王绪本, 刘艳. 基于小波变换的探地雷达信号去噪方法[J]. 电子元器件应用, 2009, 11(4): 68 −70.

[8] 郑伟, 陆广华, 陈卫东, 等. 线性度校正的新方法与系统应用[J]. 火控雷达技术, 2005, 34(4): 12 −15.

行业信息

第十二届工业仪表与自动化学术会议征文启事

由中国仪器仪表学会过程检测控制仪表分会、中国仪器仪表学会可靠性工程分会、中国自动化学会仪表与装置专业委员会、中国仪器仪表行业协会自动化仪表分会和《自动化仪表》编辑部联合主办的第十二届工业仪表与自动化学术会议将于2011年6月在上海召开。热忱欢迎广大科技、管理和应用工作者及大专院校师生等积极撰写论文,踊跃参加本届学术会议。学术会议论文投稿邮箱:iiac@sipai.com。