

中华人民共和国国家军用标准

FL 0112

GJB/Z 102A—2012

代替 GJB/Z 102—1997

军用软件安全性设计指南

Guide for military software safety design

2012—07—24 发布

2012—09—01 实施

中国人民解放军总装备部 批准

目 次

前言	II
1 范围	1
2 引用文件	1
3 术语、定义和缩略语	1
3.1 术语和定义	1
3.2 缩略语	2
4 一般要求	2
4.1 软件安全性工作通用要求	2
4.2 外购(协)或重用软件的要求	2
4.3 工具的验证要求	3
5 详细要求	3
5.1 概述	3
5.2 软件需求分析	3
5.3 软件设计	5
5.4 软件实现	16
附录 A (资料性附录) 软件安全性等级的确定方法	21
附录 B (资料性附录) 现货软件检查单示例	24
附录 C (资料性附录) 软件故障树分析(SFTA)	26
附录 D (资料性附录) X86 汇编语言安全性编码原则	31
附录 E (资料性附录) 51 系列汇编语言安全性编码原则	37
附录 F (资料性附录) 中断分析检查表	42
参考文献	44

前 言

本指导性技术文件代替 GJB/Z 102—1997《软件可靠性和安全性设计准则》。与 GJB/Z 102—1997 相比,本次修订主要有如下变化:

- a) 名称改为《军用软件安全性设计指南》;
- b) 增加了软件安全性工作通用要求、外购(协)或重用软件的要求、工具的验证要求;
- c) 增加了软件需求分析、软件设计和软件实现等三个活动中安全性有关要求;
- d) 增加了中断设计、通讯设计、自检查设计、异常保护设计、指针使用等方面内容;
- e) 对接口设计、简化设计、余量设计、防错设计、多余物处理、编码原则等方面内容进行了修改和完善;
- f) 删除了 GJB/Z 102—1997 的附录 A 和附录 C,增加了附录 B~附录 G, GJB/Z 102—1997 的附录 B 纳入了 GJB/Z 102A 的附录 A。

本指导性技术文件的附录 A~附录 F 为资料性附录。

本指导性技术文件由总装备部电子信息基础部提出。

本指导性技术文件起草单位:总装备部电子信息基础部标准化研究中心、航天科工集团公司第 706 所、船舶重工集团公司第 716 所、航天科技集团第一研究院 12 所、航天科技集团公司第 710 所、航空工业集团公司第 618 所。

本指导性技术文件主要起草人:王 伟、潘 华、石 柱、李素民、潘冠华、解月江、王 勇、王晓玲、梁 敏、段锐宁、郑 重、姚 苏。

本指导性技术文件于 1997 年 11 月首次发布。

军用软件安全性设计指南

1 范围

本指导性技术文件规定了军用软件安全性设计的实施指南。

本指导性技术文件适用于军用软件需求分析、设计和实现时的软件安全性设计，软件可靠性设计也可参考本指导性技术文件。

2 引用文件

下列文件中的有关条款通过引用而成为本指导性技术文件的条款。凡注日期或版次的引用文件，其后的任何修改单(不包含勘误的内容)或修订版本都不适用于本指导性技术文件，但提倡使用本指导性技术文件的各方探讨使用其最新版本的可能性。凡不注日期或版次的引用文件，其最新版本适用于本指导性技术文件。

GB/T 11457 软件工程术语

GJB 900 系统安全性通用大纲

GJB 2786A—2009 军用软件开发通用要求

GJB 5000A—2008 军用软件研制能力成熟度模型

GJB/Z 1391—2006 故障模式、影响及危害性分析指南

3 术语、定义和缩略语

3.1 术语和定义

GB/T 11457 确立的以及下列术语和定义适用于本指导性技术文件。

3.1.1 危险 **hazard**

可能导致事故的状态。

3.1.2 危险可能性 **hazard probability**

某种危险发生的可能程度。

3.1.3 危险严重性 **hazard severity**

某种危险可能引起的事故后果的严重程度。

3.1.4 风险 **risk**

某一特定危险事件发生的可能程度和后果严重程度的综合度量。

3.1.5 安全性 **safety**

产品所具有的不导致人员伤亡、系统毁坏、重大财产损失或不危及人员健康和环境的能力。

3.1.6 安全关键软件 **safety critical software**

其错误可能导致系统严重危险的软件。

3.1.7 安全关键功能 **safety critical function**

针对特定的危险事件，为达到或保持受控设备的安全状态而实现的功能。

3.1.8 软件失效 **software failure**

软件系统丧失完成规定功能的能力的事件。

3.1.9 软件安全性 **software safety**

软件运行不引起系统事故的能力。

3.1.10 软件安全性人员 **software safety professional**

负责或承担软件安全性工程相关任务的人员。

3.2 缩略语

CHI——人机接口
COTS——现货软件
CPU——中央处理单元
CSCI——计算机软件配置项
CSU——计算机软件单元
DSP——数字信号处理器
EPROM——可编程只读存储器
EZPROM——电可擦除可编程只读存储器
FDIR——故障/失效检测、隔离和恢复
FMECA——失效模式、影响及危害分析
FTA——故障树分析
IMR——中断屏蔽寄存器
I/O——输入/输出
RAM——随机存取存储器
ROM——只读存储器

4 一般要求

4.1 软件安全性工作通用要求

软件安全性工作的通用要求如下：

- a) 软件安全性工作是其所属系统安全性工作的重要组成部分，由系统提出要求，依据系统要求进行检查和验证，具体要求见 GJB 900。
- b) 软件安全性工作贯穿于软件生存周期全过程，应与软件工程过程活动紧密结合地进行。安全关键软件的开发应按照 GJB 2786A-2009 的要求进行，同时进行软件安全性策划，开展软件安全性工程活动。软件开发人员应参与软件安全性需求的拟定，并负责实施软件项目策划规定的有关活动，确保所开发的软件产品满足系统的安全性需求。
- c) 确定计算机软件配置项(CSCI)的安全性等级。在系统分析和设计阶段，系统人员根据系统危险分析结果，确定软件的安全性等级，软件开发人员应参与并提出建议。确定软件安全性等级的方法参见附录 A。软件开发人员应依据软件安全性等级确定软件安全性工作的具体范围。
- d) 对已纳入配置管理的受控软件的更改应进行影响域分析，并关注软件更改对系统安全的影响，特别关注软件的时序关系和存储容量的变更影响。影响域分析包括对现有功能、性能，CSCI 外部及内部的影响。其中，CSCI 外部影响主要从硬件接口、软件接口及协议、相关软件文档等方面进行分析；CSCI 内部影响主要从软件结构及调用关系、输入输出关系、数据结构、性能、内部中断、相关软件文档等方面进行分析。影响域分析结果应经过审核和批准，必要时进行评审。

4.2 外购(协)或重用软件的要求

安全关键软件采用外购(协)软件或重用软件时，安全性要求如下：

- a) 决定重用某软件来完成安全关键功能之前，应确定其适用性，并充分分析其安全性影响。附录 B 提供了一个检查单示例，以帮助进行分析。在软件开发过程中，应对重用软件产品进行安全性分析和评价工作，并对其进行验证，确保其不存在不可接受的安全性风险。
- b) 外购(协)软件应按照 GJB 5000A-2008 中供方协议管理过程域的要求进行管理，并进行安全性分析和评价。

4.3 工具的验证要求

安全关键软件开发过程中使用的影响可执行代码的软件工具，应经过安全性验证：

- a) 验证或证明软件工具的安全性与待开发的安全关键软件的安全性等级相适应；
- b) 依据软件工具使用说明，验证开发过程中所使用软件工具功能和性能的正确性、一致性和完整性。

5 详细要求

5.1 概述

5.2~5.4 提供了软件需求分析、软件设计和软件实现时的软件安全性设计的指南。

5.2 软件需求分析

5.2.1 软件安全性需求的来源

软件安全性需求是为了保护该软件所属系统的安全而对软件提出的需求，软件安全性需求的来源可分为两类：

- a) 专用的安全性需求：是所属系统特定的软件安全性需求，主要由系统向下传递的，或者是系统初步危险分析结果中危险原因与软件相关的安全性需求，其中也可能有少数是软件开发过程中新识别再由系统采纳下达的。
- b) 通用的安全性需求：是某一应用领域或多个类似的项目常常有一些共同的安全需求。为了避免工作的重复和内容遗漏，可以将多个类似项目收集起来的安全性需求整理形成特定应用领域的通用安全性需求，这种通用安全性需求会随着技术进化或新应用的实现而更新。

5.2.2 软件安全性需求开发与分析方法

5.2.2.1 概述

系统特定的软件安全性需求的开发与分析应依据系统安全性需求、环境需求、标准、项目专用规范、工具或者设施需求、接口需求、系统危险报告和系统危险分析报告，查找安全关键功能，确定安全性需求，并应根据危险发生的可能性和严重性对需求的安全关键性进行排序。需求规格说明中应明确标识软件安全关键功能。对于比较简单的软件，可以采取简单分析的方法，找出安全关键功能，对于比较复杂的软件需要采用专门分析方法找出安全关键功能，常用的有基于软件功能的故障树分析(FTA)方法和软件失效模式、影响及危害分析(FMECA)方法。软件功能的故障树分析(FTA)方法及方法应用示例参见附录 C，软件 FMECA 方法见 GJB/Z 1391-2006。

针对安全关键功能，应重点考虑如下内容，并提出安全保证措施要求，其中安全保证措施包括采用冗余、降级处理、故障处理与恢复、降级运行等。

- a) 安全运行模式、运行状态与安全条件；
- b) 容错和容失效；
- c) 危险命令处理；
- d) 接口；
- e) 数据；
- f) 定时、吞吐量和规模等。

5.2.2.2 安全运行模式、运行状态与条件

软件安全性需求应包括有效的运行模式或者状态，以及禁止或不适用的模式或者状态，应避免软件进入禁止或不适用的模式或状态。允许的参数边界可能会随着运行模式或者任务阶段的不同而不同。例如，与系统空转时相比，在试验运行期间温度可能更受限制。

在整个软件需求中查找是否存在可能导致不安全状态的条件和潜在失效隐患，例如不按顺序、错误的事件、不适当的量值、不正确的极性、无意的命令、环境干扰造成的错误、以及指挥失灵模式之类的条件，应对所有的不安全状态的条件和潜在失效隐患，制定适当的响应要求。

5.2.2.3 容错和容失效

容错系统是为了处理最有可能发生的一些故障，以及那些发生概率小但是危险的故障，以防止软件或系统进入失效状态。容错机制能防止大多数微小的差错传播演变成失效。容失效系统，忽略大多数故障，仅对较高层次的可能导致系统失效的差错进行处理。

在软件需求分析阶段应明确系统是否应能容错，或容失效，或两者兼而有之。大多数系统采用容失效(而不是容错)来达到可接受的安全性等级。对于安全关键的系统，应具有一定的容错能力，及容失效能力。

在大多情况下，系统完全关机是不合适的，如果出现太多故障或者发生非常严重的失效，宜让系统以有序安全的方式自动关闭。

冗余设计是容错和容失效的主要方法，应考虑：

- a) 依据失效容限，确定冗余要求。一般依据软件安全性等级，确定软件的失效容限要求，例如要求容许有一个或两个失效而不导致系统危险；根据软件的失效容限要求，确定软件冗余要求。对有冗余要求的软件，可采用恢复块技术、屏蔽技术或N版本程序设计技术实现。
- b) 信息冗余要求。对于安全关键信息(包括重要程序和数据)应该保存到两个或更多个存储空间，对于关乎安全关键功能的重要信息应通过两个或更多的产生方式和传输方式进行产生、收集和表决判断。
- c) 故障检测、隔离和恢复。应规定故障检测、隔离和恢复机制。
- d) 冗余管理/转换逻辑。对于要求冗余的任何危险控制，应规定冗余策略(如冷备份、热备份)和转换逻辑。

5.2.2.4 危险命令处理

危险命令是其执行(包括无意执行、失序的执行或不正确执行)能导致一个已标识的严重的或灾难性的危险，或能导致对危险控制能力降低的命令。危险命令处理包括接收、传送或者启动关键信号或者危险命令的硬件或者软件功能，危险命令处理特别注意较长的命令路径会由于通信线路电磁干扰、设备故障或(尤其是)人员差错，将增大出现不希望或不正确的命令响应的概率。

5.2.2.5 接口

接口类型包括功能接口、物理接口、人机接口。应对接口特性进行分析，分析接口出错方式及出错概率，并以此为基础确定通信方法、数据编码、错误检查、同步方法以及校验和纠错码方法。接口错误检查或者纠正措施应适合于接口的出错概率。

接口数据的定义应明确物理层和逻辑层协议，其中，

- a) 物理层协议包括：传输协议、传输率、误码率、同步通讯或异步通讯、基本数据格式(起始位、数据位、校验位、停止位)。
- b) 逻辑层协议包括：通讯数据量、帧格式、数据内容、换算要求、字节之间和帧之间的时间关系等。

在通讯接口数据定义时，还应确保通讯双方的通讯协议的一致性和完整性。

5.2.2.6 数据

应定义软件所使用的各种数据，包括规定静态数据、动态输入输出数据及内部生成数据的逻辑结构，列出这些数据的清单，说明对数据的约束；规定数据采集的要求，说明被采集数据的特性、要求和范围。对重要的数据在使用前后都要进行检验。宜建立数据字典，说明数据的来源、处理及目的地。

5.2.2.7 定时、吞吐量和规模

对于安全关键功能，应考虑系统资源和时间约束条件，进行定时、吞吐量和规模分析，分析与执行时间、I/O数据速率和内存/存储器分配有关的软件需求，考虑余量。典型的约束需求包括临界时间、自动安全保护时间、采样速率、内存资源等。

- a) 临界时间。安全关键的系统“临界时间”是从故障产生时刻到系统达到不安全状态时刻的时间

段。安全保护和恢复措施的设计应充分考虑真实世界的条件和相应的临界时间。仅当在最长的响应时间和临界时间之间有足够的余量时，自动安全保护才是有效的危险控制方式。

- b) 自动安全保护。在临界时间比现实人工操作者的响应时间短，或控制回路中没有人工参与的情况下，常常要求自动安全保护。这可以由硬件或软件或两者的结合来执行。
- c) 采样频率与物理参数的变化速率。采样速率的选择应考虑噪声水平、系统和物理参数变化特性。测量非关键信号时，采样速率应至少是最大预期信号频率的两倍；对于关键信号，以及用于闭环控制的参数，通常认为采样速率必须比系统特征频率至少高出 10 倍。
- d) 内存的使用与可用性。可以根据原先的软件开发经验来评估内存的使用，估计存储在内存中的代码的规模、存储数据空间和用于存储中间计算结果和最终计算结果的临时空间（堆、栈的规模），在安全关键软件中，应仔细考虑动态内存分配的使用。动态内存分配问题可能由如下原因造成：未释放所分配的内存（内存泄漏）；两次释放内存（引起异常）；或者缓存溢出而覆盖代码或者其他区域。
- e) I/O 通道的使用负载与可用性。应考虑输入和输出数据量与“I/O 通道”匹配性，“I/O 通道”包括内部的硬件、过程间的通信、以及外部通信接口。应分析数据收集和安全关键数据可用性之间的资源冲突。例如，在发生失效时，I/O 通道可能因错误消息而过载，而重要消息可能因此丢失或者被覆盖，可行的解决方案包括：增加部件来获取相关的和管理较低级的错误消息，或者将错误代码通过调用例行子程序传送到能够处理该问题的级别。
- f) 执行时间与 CPU 负载和可用性。分析 CPU 负载的时间变化并确定产生峰值负载的环境，分析在高负载条件下的执行时间是否可以接受；考虑多任务所造成的定时影响，例如，消息传送延迟或者因其他任务占有而不能访问所需要的资源；考虑代码是否将从 RAM 或者 ROM 中执行，ROM 的访问速度通常较慢。
- g) 程序存储空间与可执行代码的规模。估计存储在设备（EPROM、闪存等）中的可执行软件的规模，程序规模包括操作系统以及应用软件。
- h) 存储的数据量与可用的容量。考虑将产生的数据量和可用的存储空间量（RAM、磁盘等）。

5.3 软件设计

5.3.1 安全性设计的一般原则

安全关键系统的软件设计的主要目标之一是使系统安全风险最小。在可能的情况下，通过设计消除已标识的危险或降低相关的风险。降低风险的途径包括：降低软件和接口的复杂性；为用户安全而不是为用户友好而设计（但要关注操作者使用模式）；针对开发和集成过程期间可测试性的设计；给予高风险方面（危险控制等）更多的设计“资源”（时间等）。

安全性设计的基本考虑包括：

- a) 功能分配。包括：
 - 1) 确定哪些模块、类等将实现安全关键需求。尽可能将安全关键的部件与非安全关键部件隔开。
 - 2) 最小化安全关键部件的数目。安全关键部件之间的接口的设计也应实现最小交互（低耦合）。
 - 3) 文档化安全关键部件在设计层次结构中的位置和功能。
 - 4) 文档化每个安全关键部件是如何追溯到初始安全性需求的，以及该需求是如何实现的。
 - 5) 详细说明安全性相关的设计和实现约束。
 - 6) 用文档记录执行控制、中断特性、初始化、同步和对部件的控制。对于高风险系统，应避免中断，因为它们可能干扰软件安全性控制。
- b) 程序接口。包括：
 - 1) 定义所有部件之间的功能接口。对于安全关键的部件，尽可能限制它们与其他部件的交互。

- 2) 标识软件内的共享数据。设计应将安全关键的数据与其他数据隔离开来,并使非安全关键部件不能访问安全关键数据。
- 3) 文档化包含安全关键数据的数据库和数据文件以及所有访问它们的部件。
- 4) 详细说明每一个接口的设计。
- 5) 标识在接口中使用的安全关键数据。
- c) 故障检测、恢复和安全保护。包括:
 - 1) 为安全关键部件制定差错检测或恢复方案。
 - 2) 考虑对语言产生的异常和意外的外部输入(如,不合适的命令或越限的测量)的响应。
 - 3) 考虑危险操作场景。应考虑防止人工差错产生、在故障变为失效前予以识别、以及降低危险发生的风险的方法。
 - 4) 考虑在操作中是否要求进行内存测试、何时运行这些测试、以及这些测试能否影响安全关键功能。
 - 5) 考虑使用内存利用率校验对逼近的内存饱和提前告警。
 - 6) 安全保护和恢复措施的设计应充分考虑真实世界的条件和相应的临界时间。仅当在最坏(长时间)响应时间和最坏(短时间)临界时间之间有足够的裕量,自动安全保护才能够是有效的危险控制。
 - 7) 在临界时间少于现实操作人员响应时间、或在操作回路中没有人工干预时,常常需要自动安全保护。自动安全保护可由硬件或软件或两者的结合来完成,取决于实现自动安全保护的最佳系统设计。
 - 8) 在保护关键内存块数据避免遭受无意破坏或删除方面,带内存管理单元的处理器提供一种保护机制。在使用物理(如 RAM)地址或逻辑内存地址的系统上对照关键内存地址对动态分配例行程序返回的地址范围进行检查。注意逻辑和物理地址是不可相互比较的。CRC 值或差错纠正代码是对可能被意外破坏的关键数据进行检测和/或纠正的软件方法。
 - 9) 考虑必须实现哪种容错和/或容失效措施,以及实现方法;考虑是否使用多版本软件以及如何保证这些版本的功能独立性。
- d) 继承的或重用的软件和现货软件(COTS)。包括:
 - 1) 考虑对 COTS、继承或重用的软件进行的危险分析,以及对这类软件进行分析、测试和验证的信息;
 - 2) 针对这些软件遗漏的和额外的功能性设计的应对措施;
 - 3) 在文档中记录该软件在哪里使用,以及它和安全关键部件的关系。
- e) 性能和余量。包括:
 - 1) 表明安全关键部件是如何对安全性需求作出响应的,并定义这些部件的设计余量。
 - 2) 采样频率的选择应考虑噪音水平和控制系统以及物理参数的预期变化。对于非关键信号的测量,采样频率应至少是最大预期信号频率的两倍。对于关键信号,以及用于闭环控制的参数,采样频率应比系统特征频率高出至少十倍。
 - 3) 数字化系统选择的字长应至少能减少量化噪音影响以确保系统的稳定性。字长和浮点系数的选择应适合整个系统环境中处理的参数。太短的字长可能造成误解。太长的字长可能造成特别复杂的软件和 CPU 资源的过度滥用,以及调度和计时的冲突等。
 - 4) 计算机读数据、计算和输出结果需要一定的时间,因此有些控制参数将总是存在延迟。控制系统应适应这种情况,并应校验计时时钟基准数据、同步和精确性(抖动),分析任务调度。
- f) 可追踪性。包括:
 - 1) 对于每一个部件,标识到软件需求的可追踪性,尤其是到软件安全性需求的可追踪性。

- 2) 所有的需求都应向下传递到设计。维护可追踪性矩阵或其他文档以帮助对此进行验证。
- 3) 为每一个安全关键的设计特征标识测试和/或验证方法。
- g) 可测试性。包括：
 - 1) 可测试性好的设计。包括能对部件的内部进行充分测试的方法，以验证它们正确地工作；
 - 2) 应对原型代码的初步测试结果进行评价，并记录在软件开发文件中；
 - 3) 发现的任何安全关键问题都应报告给安全性人员，以帮助制定可行解决方案。

5.3.2 配合硬件或系统设计的考虑事项

配合硬件或系统设计应考虑：

- a) 电源失效防护。软件要配合硬件处理在加电的瞬间电源可能出现的间歇故障，避免系统潜在的不安全初始状态；在电源失效时提供安全的关闭；在电源的电压有波动时，使它不会产生潜在的危险。
- b) 加电检测。软件设计应考虑在系统加电时完成系统级的检测，验证系统是安全的并在正常地起作用；可能时，软件应对系统进行周期性检测，以监视系统的安全状态。
- c) 电磁干扰。对于电磁辐射、电磁脉冲、静电干扰，以及在太空中使用的计算机可能遇到的宇宙重粒子的冲击，硬件设计应按规定要求将这些干扰控制在规定的水平之下，软件设计要使得在出现这种干扰时，系统仍处于安全状态。
- d) 系统不稳定。若某些外来因素使系统产生不稳定，不宜继续执行指令，软件应采取措施，等系统稳定后再执行指令。例如，具有强功率输出的指令所引发的动作对系统或计算机系统的稳定性有影响，软件应使计算机在该指令输出并等系统稳定后，再继续执行指令。
- e) 接口故障。应充分估计接口的各种可能故障，并采取相应的措施。例如，软件应能识别合法的及非法的外部中断，对于非法的外部中断，软件应能自动切换到安全状态。反馈回路中的传感器有可能出故障并导致反馈异常信息，软件应能预防将异常信息当作正常信息处理而造成反馈系统的失控。同样，软件对输入、输出信息进行加工处理前，应检验其是否合理(最简单的方法是极限量程检验)。
- f) 干扰信号。对被控对象的变化信号中伴随存在的干扰信号采用数字滤波器加以过滤时，采样频率的确定不仅要考虑有用信号的频率，而且要考虑干扰信号的频率。
- g) 错误操作。软件应能判断操作员的输入操作正确(或合理)与否，在遇到不正确(或不合理)输入和操作时拒绝该操作的执行，并提醒操作员注意错误的输入或操作，同时指出错误的类型和纠正措施。
- h) 机械限位控制。对具有机械安全限位要求的硬件设备，软件应对输入给硬件的控制数据进行极限限位和容错处理，比如火炮的射击限位和传感器天线转动的限位的控制处理等，以避免不当的输入信息造成系统安全问题或设备故障。

5.3.3 容错和容失效的设计

为了防止故障在软件中传播，安全关键的部件应完全独立于非安全关键的部件，还应能够既检测出自身内部的错误，又不允许将错误传递下去。换言之，接收部件能够捕获并包容该错误。

- a) 围绕故障和失效的设计应考虑：
 - 1) 必须工作的功能。必须工作的功能通过独立的并行冗余实现容失效。为此，必须在各个并行路径中存在相异的软件。若两个并行串是独立的，就不可能出现能同时使两个串都失去工作能力的单一失效。若三个并行串是独立的，就不可能出现能同时使所有三个串都失去工作能力的两个失效。
 - 2) 必须不工作的功能。必须不工作的功能通过多个独立的串联禁止来达到容失效。对于那些视为独立的串联禁止而言，它们(通常)应受控于包含相异软件的不同处理器。对于独立的2个串联禁止而言，没有任何一个单一失效、人员差错、事件或者环境可以激活这2个禁

止。对于独立的3个串联禁止而言,没有任何2个失效、人员差错、事件或者环境(或者任何2个单项的组合)可以激活所有3个禁止。一般说来,这意味着每一个禁止都应由具有不同软件(例如,N-版本程序设计)的不同处理器来控制。

- 3) 故障/失效检测、隔离和恢复(FDIR)。故障/失效检测、隔离和恢复是一个尚未彻底解决的领域,其中不正确的设计能够导致系统报虚警、“伪”系统失效、或者不能检测到重要的安全关键系统的失效。在确定故障/失效检测、隔离和恢复设计时,既要考虑可能的后果,也要考虑其潜在的利益。分区和保护的原则:分区是在功能上独立的计算机软件部件之间提供隔离的技术,以确定和/或隔离故障。设计划分保护时要考虑系统的硬件资源(处理器、存储设备、输入/输出设备、中断和定时器)、控制耦合(外部存取易损性)、数据耦合(共享或重复占位数据,包括堆栈和处理器寄存器)、以及与保护机制相关的硬件设备的失效模式,以确保它们不妨碍该保护。无论保护是通过硬件还是通过软件和硬件组合来实现的,都应表明软件部件是如何划分的,包括划分的软件部件之间允许的交联的程度和范围。

b) 冗余和容错设计应考虑:

- 1) 屏蔽。对于非离散的连续变化的安全关键的参数,一种有用的冗余技术是“屏蔽”。“屏蔽”有两种方式:一种方式是,一个较高级的过程仿真一个或者一些较低级的过程,以便预测预期的性能,并判定失效是否已经在较低级的过程中发生,较高级的过程在检测到某种偏离时实施适当的冗余切换。另一种方式是,在没有足够的冗余来保持系统充分运行时,较高级的过程可以转换到某个功能子集或者降级功能集,以执行最少的功能。
- 2) 机内测试。有时故障/失效检测、隔离和恢复可以基于较低级处理器的机内测试,其中较低级的一些单元进行自我测试,并向较高级的处理器报告它们的好/坏状态。较高级的处理器将被报告为失败或不健康状态的单元关掉。
- 3) 多数表决。有些冗余方案基于多数表决,在诊断失效的准则复杂时,这个技术特别有用。(例如,当不安全的条件由超过一个模拟值而不是简单的二进制值来定义时)。多数表决要求更多的冗余来达到某个给定的容失效等级,具体如下:3取2达到容一个失效;5取3达到容两个失效。为了达到多数表决,要求奇数个并行单元。
- 4) N-版本程序设计。N-版本程序设计可以用于实现容失效的行为。软件的多个独立版本同时执行,如果结果完全一致,那么过程继续;如果存在不一致,那么使用某种表决方法来确定哪种结果是正确的。要确保达到N-版本程序设计有效的独立性,应选用各种不同的实现手段和方法来保证版本的强制相异,以减少共因故障。还可以对每一版本运算的结果增加一个简单接受测试或定时约束的功能,来先期取消被证明是错误的结果或迟迟不能到达的结果,以提高表决器的实时性和成功率。

注:仅依靠不同的设计队伍并对其他设计因素不作强制要求而开发的软件称为随机相异软件;不仅依靠不同的设计队伍,而且对方法、手段、工具、模型、语言(或语言的子集)作出强制规定而开发的软件称为强制相异软件。运行相同操作系统的两个处理器既不相相互独立,也不相互容失效。

- 5) 故障封锁区域。建立故障封锁区域,以防止软件故障的传播。应使用诸如防火墙或者“来源”检查之类的技术来为故障封锁区域提供充分的隔离,以防止危险的故障传播。最好采用硬件对故障封锁区域进行分割或者采用硬件作为故障封锁区域的防火墙。“逻辑的”防火墙可以用来隔离软件部件,例如,将一个应用程序与某个操作系统相隔离。在某种程度上,这可利用防错程序设计技术和内部软件冗余(例如,使用授权的代码或者密钥)来完成。然而,这种软件/逻辑防护措施可能因硬件失效或者电磁干扰/辐射效应而失效。获得故障封锁区域之间的独立性的典型方法是,将这些故障封锁区域驻留在不同且独立的硬件处理器上。有时,将独立的故障封锁区域驻留在相同的处理器上是可以接受的,这主要取决于特定的硬件配置(例如,故障封锁区域被存储在单独的内存芯片上,并且它们既不同时也

不并发地在相同的 CPU 上进行多任务处理)。

- 6) 冗余的体系结构。冗余的体系结构指具有两个版本的运行代码。与 N-版本程序设计不同,这两个版本并不需要等同地运行。主软件是高性能的版本,是需要运行的“常规”软件,它满足要求的所有功能和性能需求。将控制提供给“高保证”内核(也称为安全性内核)。该高保证内核可以具有与高性能软件相同的功能性,或者可以具有较为有限的范围。其主要特点在于它是安全的。该高保证内核几乎肯定是较少优化的(较慢、更容易受压、对它能够处理的载荷有较少的限制等等)。
- 7) 恢复块。恢复块使用多个软件版本以发现故障,并从故障中恢复。对一个版本的输出通过验收测试进行检查。如果它失败,那么另一个版本计算该输出,并且该过程继续。每一个后继版本更为可靠,但效率更低。如果最后一个版本失败,程序必须确定某些失效安全的方式。
- 8) 资源丰富性。资源丰富性强调通过多种方法达到系统目标。例如,如果目标是要将一个扫描平台指向某个特定目标,而并没有规定准确的运动情景,无论是移动+10°还是-350°,它都指向同一个位置,它要求系统是功能丰富的,以便提供达到目标所需要的选择。
- 9) 部件自保护和检测。部件应是自我保护和自我检测的。一个自我保护的部件不允许其他部件来毁坏它;相反,它向调用部件返回一个错误的指示。一个自我检测的部件应能检测它自己的错误,并试图从错误中恢复。自我检查是动态故障检测的一种类型,由其他技术(例如, N-版本程序设计和恢复块)使用。自我检测的种类包括复制检查(如果数据被认为是正确的,那么这些拷贝必须是相同的)、合理性检查(根据系统中的其他数据来确定数据是否合理)以及结构检查(部件是否正确地处理复杂的数据结构)。

5.3.4 接口设计

5.3.4.1 与硬件相关的接口软件设计

与硬件相关的接口软件设计应考虑:

- a) 反馈回路。系统硬件的反馈回路设计,应保证软件不可能由于反馈传感器失效引起失控条件。在软件设计中应考虑已知的部件失效模式,并将检查手段设计到软件中以检测失效。
- b) 接口控制。与安全关键硬件的接口应在所有时刻受控,即应监视该接口以确保错误或虚假数据不会意外地影响该系统,接口的失效得到检测,并且在上电、电源波动和中断、以及系统错误或硬件失效事件情况下接口安全。
- c) 判定语句。安全关键计算系统功能中的判定语句应不依靠全一或全零的输入,特别当这个信息来自外部传感器时。
- d) CPU 间的通信。CPU 间的通信应在传输安全关键数据之前成功地通过对两个 CPU 的验证检查。应进行定期检查以确保接口的完整性。检测出的错误应予以记录。如果接口多次连续传输失败,应向操作员报警,并终止安全关键数据的传输直到能进行诊断性检查。
- e) 数据传输报文。数据传输报文应是预先规定的格式和内容。每次传输应包含一个指示报文长度(如果可变的话)、数据类型或报文内容的字或字符串。至少应使用奇偶校验检查及累加和来验证数据传输的正确性,可能时应使用 CRC。在验证数据传输正确性之前任何来自数据传输报文的信息都不得使用。
- f) 外部功能。要求两个或多个来自软件的安全关键信号的外部功能(例如,点火安全设备或武器点火设备的解除保险和空中发射武器的释放)应不从单个输入/输出寄存器或缓存器接受全部必要的信号。
- g) 输入合理性检查。对于所有模拟和数字输入和输出,应在按照这些值执行安全关键功能之前进行范围和合理性检查,包括时间范围、依从关系。应根据经过验证的安全关键的模拟或数字输入执行安全关键功能。

- h) 满刻度表示(量程边界表示)。软件设计时,应使软件的满刻度和零表示都与任何数字到模拟、模拟到数字、数字到同步、和/或同步到数字转换器完全兼容。

5.3.4.2 软件模块间接口设计

设计软件模块间接口时应确保:

- a) 模块的参数个数与模块接受的输入参数个数一致;
- b) 模块的参数属性与模块接受的输入参数属性匹配;
- c) 模块的参数单位与模块接受的输入参数单位一致;
- d) 模块的参数次序与模块接受的输入参数次序一致;
- e) 传送给被调用模块的参数个数与该模块的参数个数相同;
- f) 传送给被调用模块的参数属性与该模块参数的属性匹配;
- g) 传送给被调用模块的参数单位与该模块参数的单位一致;
- h) 传送给被调用模块的参数次序与该模块参数的次序一致;
- i) 调用内部函数时,参数的个数、属性、单位和次序一致;
- j) 不得修改只是作为输入值的参数;
- k) 全局变量在所有引用它们的模块中都有相同的定义;
- l) 不存在把常数当作变量来传送的情况。

注:软件模块可以是软件单元或类。

5.3.4.3 人机界面设计

人机界面设计应考虑:

- a) 显示界面。设计时应考虑下列问题:
 - 1) 向操作员提供的的安全关键显示信息、图标、及其他人机交互方式应清晰、简明且无二义性。
 - 2) 显示应考虑颜色、字体大小和位置等因素,符合人机工程要求;
 - 3) 基于任务需求将信息分配到不同的格式或者页面;
 - 4) 对于包含在不同页面的所有必要信息相互一致;
 - 5) 页面的显示内容不宜太多。
- b) 人机接口(CHI)。设计时应考虑下列问题:
 - 1) 明确地阐述 CHI 设计的安全关键方面,包括对预想的单个或多个操作员失效的分析;
 - 2) 进行人的因素、人类工程学和认知科学的分析;
 - 3) 确保无效的操作请求被加上标记,并向操作员指明;
 - 4) 要求最少两条独立的命令来执行安全关键功能,并考虑在启动任何安全关键指令序列之前是否要求一个操作员响应或授权;
 - 5) 避免在操作员未知的情况下改变系统的安全状态;
 - 6) 安全关键状态变更时,确保有状态变更报告;
 - 7) 能清晰区别关键输入,检查输入的范围和合法性;
 - 8) 允许撤销和恢复:行动应能撤销,错误应能恢复;
 - 9) 提供适当且及时的反馈:如果操作完成,则应给出指示;如果将出现进一步的选项或者行动,则也应说明之;应使操作员能够感觉到对系统的控制以及系统对其行动的响应;
 - 10) 提供表明软件正在运行的实时指示;
 - 11) 需要若干秒或更长时间的处理功能,在处理期间应向操作员提供状态指示。
- c) 安全状态恢复。人机交互软件要便于操作员用单一动作处理当前事务,使系统退出潜在不安全状态,并恢复到某一安全状态。该动作可以包含同时按两个键、按钮或开关。在操作员反应时间不足以防止灾祸时,软件应将系统回到一个已知的安全状态,报告该失效,并向操作员报告该系统状态。

- d) 安全关键操作启动。启动安全关键操作时，应由两个或多个人员在“与”方式下操作，并有完善的误触发保护措施，以避免造成无意激活。例如，在启动某个安全关键功能时，最少应由两个不同按键来启动，并且在设计这两个按键时，应使这两个按键在操作键盘或操作面板上保持一定的距离，以免误触发。两个操作员应独立地、最好在不同的操作键盘或操作面板上同时启动，且一个操作员不能同时启动也不能采取措施强迫另一操作员启动。
- e) 误操作防护。软件应能检测不正确的操作员录入或操作，并防止由于该差错的结果而执行安全关键功能。对于该错误录入或操作应向操作员报警。报警应包括错误信息和纠正措施。软件还应提供对有效数据的录入，并向操作员提供可视和/或声音反馈，使操作员知道系统已经接受该动作并正在处理它。
- f) 报警设计。报警应使例行报警与安全关键的报警相区别，并应使得在没有采取纠正行为或没有执行所要求的后续行为以完成该操作的情况下，操作员无法清除安全关键的报警。提醒操作员注意不安全状态的信号应尽可能直接送到操作员接口。如果提供了一个操作员接口并已经检测到一个潜在不安全状态，应提醒该操作员注意检测到的异常情况、所采取的措施、以及所导致的系统配置和状态。
- g) 应考虑对故障报警记录字进行专门设计。在系统实时运行记录字的设计中，对其各分系统运行状态的故障报警标识设计，应作为一专题进行研究。对各分系统的故障报警标识设计不能只是正常、报警、故障等笼统的标识。应在各分系统的 FMEA、FTA 工作的基础上，力图给出尽可能详细的、能定位到具体故障源的故障报警标识。在尚未进行程序设计时，故障报警标识的表示位要事先留有充分的余地，在进行了详细的程序设计之后，要尽量利用故障报警标识将各种可能的故障定位。例如，对各种可能的分系统回应信息超时的具体情况（在程序中对不同分支路径）进行标识设计。当用 n 位表示故障时，其能表示的故障个数为 $2^n - 1$ ，其中用全 0 表示正常。通过记录字记录下程序运行时所有的状态和错误，便于对程序运行情况进行分析和判断，有利于故障定位等。

5.3.5 通讯设计

通讯设计时应考虑以下问题，可能时应在需求分析阶段尽早明确：

- a) 通讯接口相关联系系统界面必须协调明确。在通讯设计时，应以文件的形式明确各自任务。防止两系统间经常会因为不了解对方的任务，而使相互间的处理出现问题。
- b) 数据发送方应充分考虑数据接收方的数据处理能力。在制定通讯协议时，数据发送方应充分考虑数据接收方的数据处理能力（内存、处理时序、处理时间和处理能力），在设计中应避免数据的发送方只负责发送数据，而不考虑数据的接收方能否正确接收和处理数据。
- c) 数据接收方要充分考虑发送方的各种情况。在制定通讯协议时，数据接收方要充分考虑发送方发送数据的各种情况，例如：要有全速数据传输情况下的处理能力（除非另有间隔协议），要能识别和处理重复的数据帧，对接收的错误数据帧要设计相应的错误处理功能，防止单帧数据错误导致整个通讯信道阻塞。
- d) 接口通讯协议帧格式设计要求如下：
 - 1) 在接口通讯协议设计时，应确保通讯数据帧由帧头、数据项（含帧号、帧长、数据体）、校验字节和帧尾组成。以便接收方进行数据帧的同步，以及数据有效性的校验。
 - 2) 如果是串行异步通讯，帧头应尽量采用多字节，保证帧头的唯一性，以及对帧头后帧长的准确接收。
 - 3) 帧之间的间隔通常要大于帧内字节之间的间隔，以便接收方有足够的时间对帧头进行搜索接收。
- e) 接受数据应严格按照通讯协议进行处理。数据接收方在接收数据帧时，应按照通讯协议判断帧头、帧尾和校验字节正确后才可以使数据。

f) 通讯双方交换格式设计。

- 1) 在通讯双方的交换字格式设计时,要考虑双方计算机的字长,采用方便双方交换的字格式。如中心机字长是 32 位,而信号处理机 16 位,则涉及到信号处理的交换字应以 16 位为一个独立信息单位进行定义。不要使信号处理机靠 2 个字的部分信息(如第 1 字的后 6 位和第 2 字的前 10 位)的组合来完成一个独立信息的解释。
- 2) 对交换字各位解释说明其含义时,单一位的解释说明,不仅要说明为“1”的含义,还要说明为“0”的含义;多位解释说明时,不仅要说明特定组合的含义,还要说明其他组合的含义。如 2 位组合的含义解释说明时,不仅要说明“00 表示雷达、红外不记忆”、“01 表示雷达外记忆”、“10 表示红外外记忆”,还要说明“11”表示何含义(可能是无此定义,也可能是交换字错误、但要明确说明)。
- 3) 初始状态要设计为 0 位状态。如,“00 表示雷达、红外不记忆”即应为初始状态。又如,用“01 表示 Ku 波段”、“10 表示 Ka 波段”、“00 表示不定波段”和用“0 表示 Ku 波段”、“1 表示 Ka 波段”这两种方式对初始状态的理解是不一样的。

g) 避免双口 RAM 读写冲突。通过双口 RAM 进行信息交换是经常采用的一种设计方案,双口 RAM 使用时要注意访问冲突。解决冲突的方式有硬件握手和软件握手之分。

h) 数据传送过程中应对安全关键数据进行加密处理,以保护数据内容。

5.3.6 数据安全性设计

数据的处理应考虑:

- a) 属性控制。任何数据都应规定其合理的范围(例如,值域、变化率等等)。如果数据超出了规定的范围,就应进行出错处理。变量的定义域应预先规定,在实现时予以说明,在运行时予以检查。应对参数、数组下标、循环变量进行范围检查。
- b) 数值运算范围控制。进行数值运算时,应注意数值的范围及误差问题。在把数学公式实现成计算机程序时,要保证输入输出及中间结果不超出机器数值表示范围。
- c) 精度控制。应保证运算所要求的精度。要考虑到计算误差及舍入误差,选定足够的数据有效位。
- d) 合理性检查。在软件的入口、出口及其他关键点上,应对重要的物理量进行合理性检查,并采取便于故障隔离的处理措施。
- e) 特殊问题。在进行数学运算时,应仔细考虑浮点数接近零时的处理方式,在可能发生下溢时,使用适当小的浮点数来替代零,以避免下溢情况发生。在含有浮点数的关系判断中,不应直接进行相等关系判断。在软件设计时应考虑某些硬件(如数字协处理器)出错的处理,对在使用这些硬件的过程中出现的异常情况进行实时恢复。

5.3.7 中断设计

中断设计应考虑:

a) 中断使用的原则。包括:

- 1) 能用查询方式的不用中断方式;
- 2) 如果使用中断,中断服务程序尽量短;
- 3) 应当屏蔽无用中断,并对无用中断设置入口并返回;
- 4) 设置中断的边缘触发或电平触发,电平触发要确保电平宽度,边缘触发要有防止毛刺的措施(软件或硬件方法);
- 5) 分析可能的误中断以及频繁中断的影响,合理采取措施;
- 6) 针对具体 CPU 中断机制,可能存在同一中断多次响应(如电平触发方式、电平宽度较宽时),软件对此应有措施;
- 7) 禁止使用中断自嵌套;
- 8) 中断服务程序应严格按照所使用 CPU 的标准框架。

b) 中断的初始化、打开和使能的注意事项, 包括:

- 1) 应严格按照“阻止→关中断→初始化→开中断→使能”的顺序进行操作;
- 2) 中断初始化时要将所需要的全部资源进行初始化设置, 如触发方式和所需要使用的变量等;
- 3) 在程序中对打开中断的位置要仔细分析, 不能提前开, 也不能滞后, 要根据系统实际流程选择合理的时机;
- 4) 对于有些 CPU (如流水机制的 DSP), 简单多次的“阻止/使能”中断, 会导致中断被永远关闭, 应严格按照规定进行“阻止/使能”。

注: 中断“初始化”包括中断源的初始化(如定时中断的定时器)、中断控制器的初始化(如 8259A)、中断向量设置等;“阻止/使能”是指 CPU 不允许/允许被中断(或称响应中断), 通常 CPU 有相应的指令实现;“开/关中断”是 CPU 设置中断控制器的中断屏蔽寄存器(IMR), 允许/不允许某些中断源信号有效。

c) 慎用中断嵌套, 使用时应注意以下事项:

- 1) 尽量不使用中断嵌套, 避免嵌套的方法如: 进入中断服务程序后关掉不希望嵌套的所有中断;
 - 2) 要充分考虑中断优先级的影响;
 - 3) 使能中断和阻止中断的语句位置要独立进行仔细分析。
- d) 避免从中断服务子程序中使用非中断返回语句返回。除特殊需要外, 一定要避免从中断服务子程序中使用跳转语句或子程序返回语句直接出去, 应当使用正常中断返回语句。
- e) 要注意对中断现场的保存和恢复, 需要参考具体 CPU 中断的标准用法。要充分考虑到中断任何时候都可能发生的特点, 保存好需要保存的现场, 并在中断服务子程序返回时正确恢复现场。如在主程序中有的程序段是禁止带符号位运算, 有的程序段是允许带符号位运算, 而中断服务子程序中需要带符号位运算, 则在中断服务子程序返回时一定要恢复到中断响应时的禁止或允许带符号位运算。
- f) 必须屏蔽不用的中断源。不使用的中断源一定要通过控制字等来进行屏蔽, 而且要将不使用的中断源编为空处理的中断服务子程序, 即只有一条返回语句。软件执行过程中会受到周围环境的影响, 例如一些干扰脉冲或电磁波的影响, 如果中断控制器的各个中断源引脚一旦出现干扰信号, 将会对整个程序的执行产生不可预知的影响。因此要对不用的中断进行屏蔽。
- g) 在系统功能寄存器设置前应关中断, 防止中断影响寄存器的设置。在进行清中断使能寄存器和设置中断屏蔽寄存器之前应关中断, 禁止其他中断影响这两个寄存器的设置。如果刚刚清除中断使能寄存器时产生一个中断申请, 会使中断使能寄存器产生置位, 并造成中断的误触发。另外, 使用专用通讯芯片如 8274、82c52 等时, 当程序对该专用通讯芯片的控制寄存器状态位进行设置时, 谨防其他中断程序对同一专用通讯芯片的同一控制寄存器状态位进行设置。
- h) 程序设计时应考虑中断的优先级。由于软件中的中断有优先级, 同时中断程序和主程序也存在相互的优先顺序, 因此这些有优先级顺序的程序间, 尽量避免对同一变量进行赋值操作, 若存在不可避免的情况, 也要进行临界区保护。
- i) 软件设计时要考虑中断处理的时序。软件中有多个中断处理时, 一定要注意各个中断处理间的时序关系, 尤其是可嵌套中断间的时序关系。

5.3.8 模块设计

模块设计时应尽量简化:

- a) 模块的单入口和单出口。除中断情形外, 模块应使用单入口和单出口的控制结构。
- b) 模块的独立性。模块的独立性, 应以提高内聚度, 降低耦合度来实现, 设计时应遵循下述准则:
 - 1) 采用模块调用方式, 而不采用直接访问模块内部有关信息的方式;
 - 2) 适当限制模块间传递的参数个数;

- 3) 模块内的变量应局部化;
- 4) 将一些可能发生变化的因素或需要经常修改的部分尽量放在少数几个模块中。
- c) 模块的扇入扇出。在设计软件时,将模块在逻辑上构成分层次的结构,在不同的层次上可有不同的扇入扇出数。模块的实际结构形态应满足下述准则:
 - 1) 模块的扇出一般应控制在 7 以下;
 - 2) 为避免某些程序代码的重复,可适当增加模块的扇入;
 - 3) 应使高层模块有较高的扇出,低层模块有较高的扇入。
- d) 模块的耦合方式。模块间耦合的方式有五类,按其优选顺序排列如下:
 - 1) 数据耦合。输入的数据元素应以显示参数的形式传给另一个模块的接口。调用模块无须知道数据如何被使用。
 - 2) 控制耦合。形成控制域的接口方式。例如:使用指定要求处理类型的变量。这时,要求发送模块在控制域中置一个值,接收模块测试此值。
 - 3) 外部耦合。模块通过检查和使用已在另一模块中定义并驻留的数据变量来接收其输入的接口方式,例如:模块间共用文件。
 - 4) 公共数据耦合。所需数据是更大数据集中一部分的接口方式,例如:公共数据结构的使用。采用公共数据耦合设计的系统中,数据流是无法跟踪的。
 - 5) 内容耦合。设计一个模块时,应了解其他模块的技术细节的接口层。例如,必须知道内部开关设置的耦合。内容耦合的模块往往可共用代码行。
- e) 模块的内聚方式。模块内诸元素关联的方式有六类,按其优选顺序排列如下:
 - 1) 功能内聚。功能内聚的模块应是全部成分均为执行单一的、相对独立的处理功能的模块。例如,正弦函数、余弦函数。
 - 2) 顺序内聚。顺序内聚的模块应是一种其中处理成分的输出用于下一成分输入的模块。这种模块是基于控制结构而组织的。例如,把编辑处理和更新主文件连在一起的模块。
 - 3) 通信内聚。通信内聚的模块应是一种全部元素在同一类型输入数据集上操作并(或)产生同样类型的输出数据的模块。这是一种将输入和输出组合在相同模块中的模块化。例如,在某专用文件上执行全部 I/O 功能的模块。
 - 4) 时间内聚。为处理面向时间的活动而建立的模块。这是一种其成分仅在时间上相关的内聚方式。例如,将初始化例程和终止例程组织在同一模块中。
 - 5) 逻辑内聚。以逻辑分组形式组合的模块。例如,打印一组在许多程序段中生成的不同错误信息。
 - 6) 偶然内聚。其中的元素没有关系或几乎没有关系的模块。例如,基于语句条数、任意把程序分段而形成的模块。

5.3.9 定时、吞吐量和规模的余量设计

定时、吞吐量和规模的余量设计应考虑:

- a) 资源分配及余量要求。在软件设计时,应确定有关软件模块的存储量、输入输出通道的吞吐能力及处理时间要求,并保证满足系统规定的余量要求,一般要求应留有不少于 20% 的余量。
- b) 时序安排的余量考虑。软件工作的时序安排,要结合具体的被控对象确定各种周期。如采样周期、数据计算处理周期、控制周期、自诊断周期、输出输入周期等。当各种周期在时间轴上安排不下时,应采取更高性能的 CPU 或多 CPU 并行处理来解决,以确保软件的工作时序之间留有余量。

5.3.10 防错设计

防错设计应考虑:

- a) 参数化。在软件设计中,应规定用统一的符号来表示参数、常量和标志,以便在不改变源程序

逻辑的情况下，对它进行更改。

- b) 标志。所有标志应进行严格的定义，并编制标志的使用说明。对于安全关键的标志，在其被使用的软件单元里，应唯一且用于单一目的。标志的内容包括：
 - 1) 名称和位定义；
 - 2) 功能和作用；
 - 3) 使用范围(有效范围)；
 - 4) 生存周期(初始状态、运行中的变化条件、状态和时刻、最终状态)；
 - 5) 使用情况(使用该标志的模块名及使用方式)。
- c) 安全关键信息。对于安全关键信息，设计时应注意：
 - 1) 安全关键信息不能仅由单一 CPU 命令产生。
 - 2) 不用寄存器和 I/O 端口来存储安全关键信息。
 - 3) 安全关键信息的表示：使安全关键信息不会因一位或两位差错而引起系统故障。安全关键信息与其他信息之间应保持一定的码距。例如，不能用 01 表示一级点火，10 表示二级点火，11 表示三级点火。不得使用一位的逻辑“1”和“0”表示，建议使用 4 位或 4 位以上、既非全 0 又非全 1 的独特模式来表示，以确保不会因无意差错而造成危险。例如，可用四位模式“0110”来表示系统的安全状态，用“1001”来表示系统的危险状态，其他模式表示系统状态出错，需要系统对其进行处理。
 - 4) 如安全关键信息有差错，应能检测出来，并返回到规定的状态(例如，安全状态)。
 - 5) 安全关键信息的决策判断依据不得依赖于全“1”或全“0”的输入(尤其是从外部传感器传来的信息)。
- d) 安全关键功能。安全关键功能的运行，应在接到两个或更多个相同的信息后才执行。
- e) 非授权存取的限制。
 - 1) 应防止对程序(源程序、汇编程序及目标代码)的非授权的或无意的存取或修改，其中包括对代码的自修改；
 - 2) 应防止对数据的非授权的或无意的存取或修改；
 - 3) 对安全关键功能模块应设置调用密码。
- f) 文件。文件应唯一且用于单一目的：文件在使用前应成功地打开，在使用结束后应成功地关闭；文件的属性应与它的使用相一致。
- g) 无意指令跳转的处理。应检测安全关键软件内或安全关键软件间的无意跳转；如果可行，应进行故障诊断，并确定引起无意跳转的原因。应提供从无意指令跳转处进入故障安全状态的恢复措施。
- h) 程序检测点的设置：
 - 1) 在安全关键软件中的关键点上进行监测，在发现故障时进行故障隔离；必要时，使系统进入安全状态；
 - 2) 在完成必要检测功能的前提下，检测点宜少不宜多；
 - 3) 测试特征量流出通道应力求独立，使测试功能的失效不会影响其他功能。
- i) 寻址模式的选用。尽量不使用间接寻址方式。在确实有必要采用间接寻址方式时，需慎重考虑和充分论证，并在执行之前验证地址是否在可接受的范围内。
- j) 数据区隔离。为了防止程序把数据错当指令来执行，要采用将数据与指令分隔存放的措施。必要时在数据区和表格的前后加入适当的 NOP 指令和跳转指令。使 NOP 指令的总长度等于最长指令的长度，然后加入一条跳转指令，将控制转向出错处理程序。
- k) 信息存储。考虑到信息保存的可靠性，对不需修改的重要信息，条件允许时应放在不易丢失的只读存储器(ROM)中。对需要少量次数修改的重要信息，则应放在电可擦除可编程只读存储

器(EZPROM)中。在宇宙空间中不得使用电可编程只读存储器(EPROM)。

- 1) 对安全关键信息,应保存在多种或多个不同芯片中,并进行表决处理;
 - 2) 对可编程只读存储器(PROM)中的重要程序进行备份(例如,备份在不同的 PROM 中),一旦 PROM 中的程序被破坏,还可通过遥控命令等手段使系统执行其备份程序;
 - 3) 对随机存取存储器(RAM)中的重要程序和数据,应存储在三个不同的地方,而访问这些程序和数据都通过三取二表决方式来裁决。
- l) 算法选择:
- 1) 对规定时间内要完成规定任务的软件,不能采用没有把握在一定时间内算出结果的算法。例如,不应采用只有“计算迭代到 $|\Delta| < \varepsilon$ 为止”的算法,而采用迭代到 $|\Delta| < \varepsilon$ 为止或迭代到一定次数为止。
 - 2) 算法所使用的存储空间应完全确定。例如,尽量不采用动态堆空间。

5.3.11 自检查设计

自检查设计应考虑:

- a) 监控定时器的设计。监控定时器设计时应遵循下述准则:
- 1) 应提供监控定时器或类似措施,以确保微处理器或计算机具有处理程序超时或死循环故障的能力。
 - 2) 监控定时器应力求采用独立的时钟源,用独立的硬件实现;若采用可编程定时器实现,应统筹设计计数时钟频率和定时参数,力求在外界干扰条件下定时器受到干扰后定时参数的最小值大于系统重新初始化所需的时间值,最大值小于系统允许的最长故障处理时间值。
 - 3) 与硬件状态变化有关的程序设计应考虑状态检测的次数或时间,无时间依据的情况下可用循环等待次数作为依据,超过一定次数作超时处理。
 - 4) 计时器复位设计,应使软件不能进入某个内部循环或作为该循环指令序列的一部分复位该计时器。适用时应使系统返回到某个已知安全状态并向操作员报警。计时器的设计必须确保主要 CPU 时钟的失效不能损害其功能。
- b) 存储器检查。必须定期检查存储器、指令和数据总线。测试指令序列的设计必须确保单点或很可能的复合失效被检测出来。加载时必须进行数据传输的检查和程序加载验证检查,并在此后定期进行,以确保安全关键代码的完整性。
- c) 故障检测。对于计算系统的安全关键子系统必须编写故障检测和隔离程序。故障检测程序必须设计成在这些有关安全关键功能执行之前检测潜在的安全关键失效。故障隔离程序必须设计成将故障隔离到实际的最低级,并向操作员或维护人员提供这个信息。
- d) 运行检查。可测试的安全关键系统元素的运行检查必须直接在执行有关安全关键操作之前进行。

5.3.12 异常保护设计

异常保护设计应考虑:

- a) 仔细分析软件运行过程中各种可能的异常情况,设计相应的保护措施;
- b) 当采用外购(协)或重用软件时,应仔细分析外购(协)或重用软件原有的异常保护措施对于现有的软件需求是否足够且完全适用;
- c) 异常处理措施应使系统转入安全状态,保存异常出现的现场信息,并保持计算机处于运行状态。

5.4 软件实现

5.4.1 概述

在软件实现阶段,安全性需求已经从设计向下传递到代码级。软件编制时应遵循相应的编制规范,编制完成后应进行代码验证工作。

5.4.2 软件编码

5.4.2.1 总则

软件编码时应遵循编程标准，以保证软件实现的安全性。其中 C/C++ 语言编码的要求可参考有关标准，X86 系列汇编语言编码的原则可参考附录 D，51 系列汇编语言的编码原则可参考附录 E。对于软件编程的通用要求见 5.4.2.2~5.4.2.7。

5.4.2.2 编程语言通用要求

编程语言的通用要求如下：

- a) 采用标准化的程序设计语言进行编程；
- b) 在同一系统中，应尽量减少编程语言的种类；
- c) 应选用经过优选的编译程序或汇编程序，应使用正版软件；
- d) 为提高软件的可移植性和保证程序的正确性，建议只用语言编译程序中符合标准的部分进行编程，避免使用编译程序引入的非标准部分进行编程。
- e) 不得使用 GOTO 语句；
- f) 对顺序程序的编制，应参见 GJB 2255 推荐的几种控制结构来编制程序；对并序程序的编制，应选择便于测试且简单的结构来编制程序；
- g) 禁止使用未初始化的变量；
- h) 注意内存的管理与使用，避免造成内存泄漏；
- i) 注意由编译程序引入的不确定的行为（如：产生无法追踪的目标代码）；
- j) 禁止程序正常执行时直接从过程中跳出。

5.4.2.3 软件复杂性控制

为了控制程序的复杂性，便于后期的测试和维护，在进行软件实现时应遵守如下原则：

- a) 模块的圈复杂度（即，McCabe 指数）一般不大于 10；
- b) 对于高级语言实现的模块，每个模块的源代码最多不超过 200 行，一般控制在 60 行以内；
- c) 每个模块所传递的参数个数一般不超过 6 个；
- d) 每个模块的扇入和扇出一般不大于 7。

5.4.2.4 注释要求与方法

注释要求与方法如下：

- a) 注释的一般要求：为提高程序的可读性，在源程序中应有足够详细的注释。注释应为功能性的，而非指令的逐句说明。人工编写的源程序中注释的行数一般不得少于源程序行数的 20%。
- b) 模块头部注释要求：在每个模块的可执行代码之前，应用一段文字注释来说明如下内容：
 - 1) 模块名注释：标识模块的名称、版本号、入口点、程序开发者的姓名、单位和开发时间，如有修改，还应标识修改者的姓名、单位和修改时间；
 - 2) 模块功能注释：说明模块的用途和功能；
 - 3) 输入/输出注释：说明模块使用的输入/输出文件名，并指出每个文件向模块输入，还是从模块输出，或两者兼而有之；
 - 4) 参数注释：说明模块所需的全部参数的名称、数据类型、大小、物理单位及用途，说明模块中使用的全局变量的名称、数据类型、大小、物理单位及其使用方式，说明模块的返回值；
 - 5) 调用注释：列出模块中调用的全部模块名和调用该模块的全部模块名；
 - 6) 限制注释：列出限制模块运行特征的全部特殊因素；
 - 7) 异常结束注释：列出所有异常返回条件及动作；
 - 8) 方法注释：说明该模块为实现其功能所使用的方法，为简练，亦可指出说明该方法的文档；
 - 9) 外部环境及资源注释：说明该模块所依赖的外部运行环境及所用资源，如操作系统、编译程序、汇编语言、中央处理机单元、内存、寄存器和堆栈等等。

- c) 模块内注释要求：在模块中，至少对有条件改变数据值或执行顺序的语句（即，分支转移语句、输入输出语句、循环语句、调用语句）进行注释，对这些语句的注释不得扰乱模块的清晰性，即这些注释也应符合程序的缩进格式。具体的注释方法如下：
 - 1) 分支转移语句：指出执行动作的理由；
 - 2) 输入输出语句：指出所处理的文件或记录的性质；
 - 3) 循环语句：说明执行动作的理由及出口条件；
 - 4) 调用语句：说明调用过程的理由及被调用模块的功能。
- d) 安全关键内容注释要求：在每个模块和软件单元(CSU)中，对关键的语句标号和数据名应有准确的引用信息，并确定所有输入值的允许和期望范围；对计数器的值的注释应包含计时功能的描述、其值及其基本原理或所引用的解释计数器值的基本原理的文档。

5.4.2.5 指针使用

指针使用应考虑：

- a) 禁止将参数指针赋值给过程指针：将参数指针赋值给过程指针会导致不可预料的结果，因此禁止将参数指针赋值给过程指针。
- b) 禁止指针的指针超过两级：对指针进行控制是很困难的，当指针的指针超过两级时，使用起来更具风险，因此禁止指针的指针超过两级，在某种特殊情况下确实需要超过两级的指针时，必须加以明确注释，在软件详细设计说明的指针特殊设计中详细说明理由，并在单元测试报告中提供针对性的测试结果。
- c) 禁止引用空指针和无效指针：引用可能为空或无效的指针，得到的结果显然是错误的，会带来意想不到的问题。
- d) 谨慎使用指针的逻辑比较：使用大于和小于的操作符对指针进行比较是具有较大风险的，应谨慎使用指针的逻辑比较。
- e) 谨慎对指针进行代数运算：对指针进行代数运算具有较大风险，应谨慎对指针进行代数运算。
- f) 谨慎将过程声明为指针类型：使用过程式指针具有较大风险，因此谨慎将过程声明为指针类型。

5.4.2.6 多余物的处理

多余物的处理应考虑：

- a) 文档中未包含内容的处理：运行和支持程序应包含文档所要求的那些特征和能力，而不应包含文档中所没有的特征。对于为便于软件测试而引入的必要功能和特征，应验证它们不会影响软件的可靠性和安全性。
- b) 程序中多余代码的清除：
 - 1) 运行程序不得包含不使用（未覆盖到需求的）的可执行代码。不使用（未覆盖到需求的）的可执行代码应从源程序及重新编译的程序中去掉；
 - 2) 装入的运行程序不得包含未使用的变量。在固化程序之前应去除不再运行的程序部分；
 - 3) 对不同运行阶段的多余代码（如：调试时使用的监控程序），应慎重考虑；
 - 4) 对于因可选择选项功能或软件重用等而引起的无效码，在软件设计中应进行描述，并进行必要的测试和分析，以保证其不影响系统的安全性。
- c) 未使用的内存处理：
 - 1) 所有未被运行程序使用的内存必须初始化成某一模式，该模式的执行将使系统恢复到确定的安全状态；不得用随机数、暂停、或等待指令填充处理器的内存；空指令、停止指令要谨慎使用；不得保留先前覆盖中的或装入的数据或代码。当处理器收到这种非执行的代码模式而暂停运行时，监控定时器必须提供一个中断执行程序来使系统恢复到安全状态。如果处理器把这些非执行代码模式视为一个错误，则应开发一个错误处理执行程序来将系统恢复到某个安全状态，并终止处理。

- 2) 未使用的内存包括程序的“空白区”和数据的“空白区”。其中程序的“空白区”应该在固化时做适当的处理，数据的“空白区”应该在系统的初始化时做适当的处理。处理的对策应根据指令系统功能、故障处理的实时性及其他要求来确定。

5.4.2.7 其他要求

其他要求如下：

- a) 程序检测点的设置：
 - 1) 在安全关键软件中的关键点上进行监控，在发现故障时应进行故障隔离；必要时，使系统进入安全状态；
 - 2) 在完成必要的检测功能的前提下，检测点尽量少；
 - 3) 测试特征量流出通道应力求独立，使测试功能的失效不会影响其他功能。
- b) 公用数据与公用变量：应指明由两个或多个模块公用的数据和公用变量，并尽量减少对公用变量的使用和改变，以减少模块间的依赖度和读写冲突。
- c) 文件的处理要求：文件应唯一且用于单一目的；文件在使用前应成功地打开，在使用后应成功地关闭；文件的属性应与它的使用相一致。
- d) 禁止程序修补：贯穿开发过程始终应禁止修补。所有软件更改应用源语言编码并在进入运行或测试设备前进行编译。在运行期间，可能需要为一个程序打补丁，但应谨慎，并执行 4.1 的规定。

5.4.3 代码验证

5.4.3.1 概述

代码验证目的在于验证软件正确地实现了经过验证的设计且不违反安全性需求。由于已经具备了源代码，因此可以实际测量软件的规模、复杂性和资源使用情况。这些量在设计阶段只可能进行估计和推测，如果代码分析的结果表明这些“估计和推测”很不正确，就可能导致重新设计。代码验证的方法包括代码逻辑分析，代码数据分析，代码接口分析，未使用代码分析，中断使用分析，定时、吞吐量和规模分析，以及代码审查等方法。

5.4.3.2 代码逻辑

代码逻辑分析检查软件中的逻辑错误。这些分析通过逻辑重构、方程式重构等来完成。对于复杂的软件，可只对其安全关键的部件进行代码逻辑分析。对于其他软件部件，如果认为其对系统功能来说是重要的，也可对这些部件进行代码逻辑分析。

- a) 逻辑重构需要从代码生成流程图，并将代码与设计说明和流程图进行比较。
- b) 通过将代码中的方程式与设计中的方程式进行比较来完成方程式重构。
- c) 即使在关键的指令序列可能伪装为数据的情况下，也应标识关键的指令序列。分析员应确定每一条指令是否有效，并确定执行该指令的条件是否有效。
- d) 宜尽可能使用自动化的工具辅助进行代码逻辑分析。

5.4.3.3 代码数据

代码数据分析的重点是分析软件中的数据结构和使用情况。数据分析应关注如何定义和组织数据项。通过将代码中所有数据项的值与设计中的描述相比较来完成代码数据分析。应特别关注安全关键数据的完整性，防止其被无意地改变或者覆盖。例如，检查中断过程是否干扰安全关键的数据；检查说明为安全关键变量的“类型”。

5.4.3.4 代码接口

代码接口分析的目的在于验证一个软件部件的内部接口和外部接口的兼容性。每一个接口都是一个潜在问题的来源。代码接口分析的目的在于验证接口已经得到了适当的实现。检查参数适当地通过接口进行了传送。验证接口各侧的数据规模、测量单位、字节序列和字节中的位次序是相同的。

5.4.3.5 未使用代码

若软件中存在未使用的代码,就可能使得软件中存在不必要的复杂性,可能占用了通常为有限资源的内存或者大容量存储器;或可能包含那些如果偶然执行(例如,由于硬件失效或者由于一次单粒子反转)可能产生危险的子程序。因此在编码期间需要使用人工方法(如代码走查)或工具检查是否存在未使用的代码,并对未使用代码进行分析。

5.4.3.6 中断使用

中断若使用不当则会对控制流和数据流产生不良影响,因此中断分析的重点是关注中断使用的正确性。如:中断是否能够导致优先级倒置或者妨碍高优先级或安全关键任务的完成?如果在一段时间内禁止中断,那么引入中断系统栈是否能够防止中断丢失?低优先级的处理是否能够中断高优先级的处理并改变关键的数据?

在进行中断分析时,应考虑:

- a) 基本情况:列举该软件用到的全部中断,但不包括软中断。分析它们属于定时中断还是周期性中断或随机中断,是偶发的还是频繁的,是系统内的还是系统外的。
- b) 中断之间的关系:检查中断优先级分配情况;中断之间是否有约束机制,如 A 中断必须在 B 中断之后出现等。
- c) 中断的处理流程:中断入口的保护;中断出口的恢复;中断执行时间是否合适;开中断的时机是否恰当。
- d) 中断的嵌套:中断控制初始化是否和硬件电路相容,是否和系统功能要求一致;可能发生的最大嵌套次数;最大嵌套时的执行时间测试和估计;最大嵌套时的堆栈使用情况;是否有自嵌套的情况;是否会丢失低级中断;是否会发生死锁。
- e) 资源竞争检查:有无共享的缓冲区、共享变量和 I/O 端口存在。读写处理是否有冲突;是否有同时出现的中断申请。
- f) 异常情况处理:是否采用了防止干扰引起中断误触发的措施(含硬件),如抑制干扰源、切断干扰通路、提高抗干扰能力等;发生误触发或丢失中断对系统功能有哪些影响。
- g) 再入代码:再入代码是为中断时不丢失状态信息而设计的。检查再入部件为每一个中断保存了足够的数据,并且数据和系统状态的恢复正确。
- h) 可中断的代码段/部件:如果定时关键区域无法接受延时,则应确保对其进行了保护而不被中断。检查不应被中断的指令序列。
- i) 空中断处理:考虑在接收到一个空中断时将会发生什么,需要如何处理。

设计人员进行中断检查时应阅读代码和相关文档,形成中断基本信息一览表,格式可参考附录 F 的表 F.1。在对中断使用情况了解的基础上,应对每一个中断处理进行分析,并文档化,其格式可参考附录 F 的表 F.2、表 F.3 和表 F.4。

5.4.3.7 代码审查

应对安全关键软件进行代码审查,代码审查单有助于提高审查效果。

5.4.3.8 定时、吞吐量和规模

编码阶段结束后,便可以测量定时、吞吐量和规模参数。可执行部件的规模(存储量的大小)容易测量,因为它就是由该运行软件所使用的内存空间量。为了确定内存的最大使用量,定时和吞吐量参数可能需要运行特定的测试,其中有些测试可能需在测试阶段进行,因为在测试阶段这些参数才正式地包含在功能测试或者负载/强度测试之中。只要相应的代码是稳定的,应尽早执行测试,以验证定时、吞吐量和规模需求。

附 录 A

(资料性附录)

软件安全性等级的确定方法

A.1 概述

确定软件安全性等级的步骤主要包括：

- a) 采用初步危险分析(PHA)的方法，确定系统潜在危险；
- b) 分析确定危险的严重性和可能性；
- c) 确定系统风险指标；
- d) 确定软件在系统中执行的控制类别；
- e) 最后确定软件安全性等级。

A.2 确定系统潜在危险

系统初步危险分析是系统中各层次开展危险分析的前提和基础，也是开展软件系统危险分析、确定软件关键等级的前提和基础。通过系统的初步危险分析，列出系统所有的潜在危险，包括危险的原因以及针对危险原因的危险控制等。

A.3 分析、确定潜在危险的严重性和可能性

危险的严重性和可能性主要基于过程的判断得到。危险的严重性可分为四级：灾难的、严重的、轻度的、轻微的，参见表 A.1。危险可能性可分为五级：经常、很可能、偶然、很少、极少，参见表 A.2。

表 A.1 危险严重性等级定义

危险严重性等级	定 义
灾难的	人员死亡，或系统报废，或基本任务失败，或环境灾难。
严重的	人员严重伤害，或系统严重损坏，或基本任务的主要部分未完成，或环境严重破坏。
轻度的	人员轻度伤害，或系统轻度损坏，或对完成任务有轻度影响，或环境轻度破坏。
轻微的	对人员的伤害和系统的损坏轻于轻度，或虽然执行任务有障碍但是能够完成，或对环境的破坏可忽略。
注：决定使命成败的任务为基本任务。	

表 A.2 危险可能性等级定义

危险可能性等级	定 义
经常	危险可能经常发生： $P > 10^{-1}$
很可能	在某一项的生命期中危险将发生若干次： $10^{-2} < P \leq 10^{-1}$
偶然	在某一项的生命期中危险可能偶尔发生： $10^{-3} < P \leq 10^{-2}$
很少	在某一项的生命期中危险发生的可能很少： $10^{-4} < P \leq 10^{-3}$
极少	危险几乎不可能发生： $P \leq 10^{-4}$
注：P 是危险发生的概率。针对不同系统，各级 P 值的划分可能有别。	

A.4 确定系统的风险指标

根据危险的严重性等级及其可能性等级，确定每种危险的系统风险指标，参见表 A.3。

表 A.3 系统风险指标

危险严重等级	危险可能性				
	经常	很可能	偶然	很少	极小
灾难的	1	1	2	3	4
严重的	1	2	3	4	5
轻度的	2	3	4	5	6
轻微的	3	4	5	6	7
注：风险指标中的 1 表示最高系统风险，7 表示最低系统风险。					

其中最高风险的危险(即系统风险指标为 1)在系统分析与设计中是不允许存在的。如果存在，系统分析与设计就应重新进行。如果风险指标为 6、7，通常可以不考虑安全性工作。

A.5 确定软件系统中每个软件的控制类别

在系统危险分析的基础上开展软件系统的危险分析，以确定每个软件的控制类别。其中有一些软件是直接导致危险的原因，而有些软件则是用来控制、缓解或检测危险的。软件系统危险分析的目的就是要将这些与软件相关的危险以及与这些危险相关的软件标识出来。软件的控制类别与软件对系统的控制程度、控制的复杂性和实时性等有关。

软件的控制类别可分为四类：I、II、III、IV，参见表 A.4。

表 A.4 软件的控制类别定义

软件控制类别	定 义	
I	1	软件对危险进行部分或全部的自主控制。
	2	包含多个子系统、交互作用的并行处理器或多个接口的复杂系统。
	3	有一些或全部安全关键功能是时间关键的。
II	1	控制危险，但是其他安全系统可以进行部分缓解；检测危险，需要采取安全措施时通知操作员。
	2	包含少量子系统和/或一些接口的中等复杂系统，无并行处理。
	3	有一些危险控制动作可能是时间关键的，具有临界时间要求，但是不会超过中等熟练程度的操作员所需要的时间或自动系统响应的时间。
III	1	若软件发生故障，则有一个或若干个缓解系统防止危险发生；或提供冗余的安全关键信息资源。
	2	稍微复杂的系统，仅包含有限数目的接口。
	3	缓解系统能在任何临界时间内进行响应。
IV	1	对危险无硬件控制，并不为操作员提供安全关键数据。
	2	仅带 2-3 个子系统的简单系统，仅包含少量接口
	3	非时间关键，没有临界时间要求
注：每个软件控制类别都有三种描述，它们之间是“或”的关系，即只要符合一种，就可以确定。		

A.6 确定软件的安全性等级

根据每个软件的控制类别和系统风险指标确定每个软件的安全性等级。软件安全性等级从高到低为 A、B、C、D 四个等级。参见表 A.5。

表 A.5 软件安全性等级矩阵

软件控制类别	系统风险指标			
	2	3	4	5
I	A	B	C	D
II	B	C	D	D
III	C	D	D	D
IV	D	D	D	D

A.7 安全关键软件

安全性等级高的软件为安全关键软件。通常 A、B 级的软件为安全关键软件。

附 录 B
(资料性附录)
现货软件检查单示例

现货软件(COTS)检查单示例见表 B.1。

表 B.1 现货软件检查单

序号	待考虑项	问题回答(是/否)	计划的措施
1 ^a	供应商的设施和过程是否经过审核? 允许对供应商的设施和过程进行一次审核。不论什么原因, 如果不能实施一次审核, 那么该 COTS 被视为具有未缓和的重大风险, 因此可能对目标设备来说该 COTS 可能是不适合的。		
2 ^a	COTS 的验证和确认活动是否合适? 证明 COTS 软件所执行的验证和确认活动是合适的且是充分的, 能满足设备的安全性和有效性要求。		
3 ^a	没有供应商的支持, 项目能否维护 COTS? 确保项目能在供应商不再提供支持后还能维护 COTS。		
4	软件是否包含接口、防火墙、封装程序等? 及早考虑接口、防火墙、封装程序和连接程序。当创建封装程序时, 应避免对内部产品接口和功能的依赖性, 或者隔离这些依赖性。		
5	软件是否提供诊断? 注意内置诊断和错误处理。		
6	有无影响软件选择的任何关键产品? 在产品评估前, 确定有无能影响选择的其他关键产品(或策略, 标准)。		
7	以前是否使用过该软件供应商的产品吗? 利用过去与供应商/产品有关的经验。寻求来自其他项目的信息。使用信息数据库, 并牢记一个产品的行为可能随着其使用方式的不同而变化。		
8	是否是初始版本? 不要购买 1.0 版本。		
9	是否调查过其竞争对手? 向该产品的竞争对手了解其类似产品。		
10	能否获得源代码? 考虑购买源代码, 以便能够自己实施测试。值得指出的是, 购买源代码的代价昂贵, 而且通常需要放弃技术支持和/或产品担保。		
11	行业标准接口是否可用? 确保产品使用行业标准接口。		
12	产品研究是否彻底? 根据对事实的分析选择产品。		
13	对现货软件驱动程序包是否进行了确认? 在系统接口确认过程中包括现货软件驱动程序包的确认过程。这包括对数据信号进行双向数据值的验证; 适用时, 对控制信号的各种模式设置进行双向验证; 以及对具有 CPU 和操作系统的驱动程序的输入/输出中断和定时功能进行验证。		

表 B.1 (续)

序号	待考虑项	问题回答(是/否)	计划的措施
14	是否存在未被使用的特征? 确定如何处理未被使用的特征。		
15	代码自动生成工具是否经过独立的确认? 确定代码自动生成工具是否经过独立的确认。现货工具选择应遵循与组成部件选择相同的过程。		
16	先前的配置能否得到恢复? 重新评价每一个版本, 并确保先前的配置能够得到恢复。		
17	是否需要为处理器进行一次重新编译? 对更换一个处理器就需要一次重新编译的系统, 进行完整而全面的重新测试。		
18	是否进行了安全性影响评估? 在将一些新的或者修改的 COTS 组成部件纳入某个基线系统时, 应进行一次安全性影响评估。在失效模式和影响分析表格中记载这些危险。确保在危险报告、设计需求和测试报告之间存在可追踪性。分析应包括对已知问题报告、用户手册、规范、补丁、文献等的评审以及对其他用户使用该 COTS 软件经验的因特网搜索。		
19	COTS 工具是否影响安全性? 在选择 COTS 工具时应牢记该工具的目的。确定其结果是否容易验证, 并确定工具的使用结果是否影响到安全性决策。		
20	COTS 的应用是否适当? COTS 产品的用途应与其被开发软件的目的一致。		
21	在 COTS 和现货硬件之间是否存在兼容性? 应认识到, 并不是所有的现货硬件都能运行所有的 COTS。		
22	供应商是否有质量体系认证? 确定供应商是否经过质量体系认证, 或者是否具有 GJB 5000/GJB 5000A、CMMI 的 3 级或更高级。这提供对他们的开发过程的信心。		
23	供应商是否从他们自己的供应商那里获得高质量产品吗? 确保供应商意识到他们应对他们承包商和转包商的产品质量负责。		
^a 有生命威胁危险的项目应检查这些项。			

附录 C

(资料性附录)

软件故障树分析(SFTA)

C.1 概述

故障树分析(FTA)是系统化、形式化的分析方法,用一套逻辑和事件符号构建故障树,直观地表示故障事件之间的逻辑关系,以演绎的方法找出导致系统故障的各种可能的基本原因,继而找到系统的薄弱环节,以改进系统的分析与设计。

软件故障树分析(SFTA)即是将 FTA 的思想与方法用于软件故障分析,是一种自顶向下的软件可靠性和安全性分析方法,即从软件系统不希望发生的事件(顶事件),特别是对人员和设备的安全产生重大影响的事件开始,向下逐步追查导致顶事件发生的原因,直至基本事件(底事件)。

SFTA 主要包括软件故障树的建立和软件故障树的定性分析。

C.2 实施步骤

C.2.1 软件故障树的建立

故障树建立的完善程度直接影响定性分析和定量分析的准确性。软件故障树的建立通常包括如下步骤:广泛收集并分析有关技术资料;选择要分析的顶事件;构建故障树;简化故障树。

软件故障树中使用的符号包括事件符号和逻辑门符号两类。事件符号用以表示故障事件,逻辑门符号用以表示故障事件之间的逻辑关系。图 C.1 中列出了几种通用的软件故障树符号。

建立软件故障树通常采用演绎法,即首先选择要分析的顶事件(即不希望发生的故障事件)作为故障树的“根”,然后分析导致顶事件发生的直接原因(包括所有的事件或条件),并用适当的逻辑门与顶事件相连,作为故障树的“节”(中间事件)。按照这种方法逐步深入,一直追溯到导致顶事件发生的全部原因(底层的基本事件)为止。这些底层的基本事件称为底事件,构成故障树的“叶”。

在建立软件故障树时,应注意如下方面:

- a) 底事件定义:一个软件在其需求分析阶段确定了各项软件功能,在概要设计阶段设计出了各软件部件,在详细设计阶段设计出了各软件单元。因此在软件开发各阶段实施 SFTA 时, SFTA 深入的层次是不同的。在需求分析阶段, SFTA 只可以深入到功能层,因此软件的功能失效即为底事件;在软件概要设计阶段 SFTA 可以深入到软件部件层,因此软件部件失效即为底事件;在软件详细设计阶段, SFTA 可以深入到软件单元层,因此软件的单元失效即为底事件。
- b) 控制流分析:软件功能都是通过某一个或若干个软件部件或软件单元的执行来实现的。程序控制流程反映这一实现过程。控制流程中的某个环节失效,便有可能软件失效。因此可以在对软件控制流程的分析的基础上建立软件故障树。

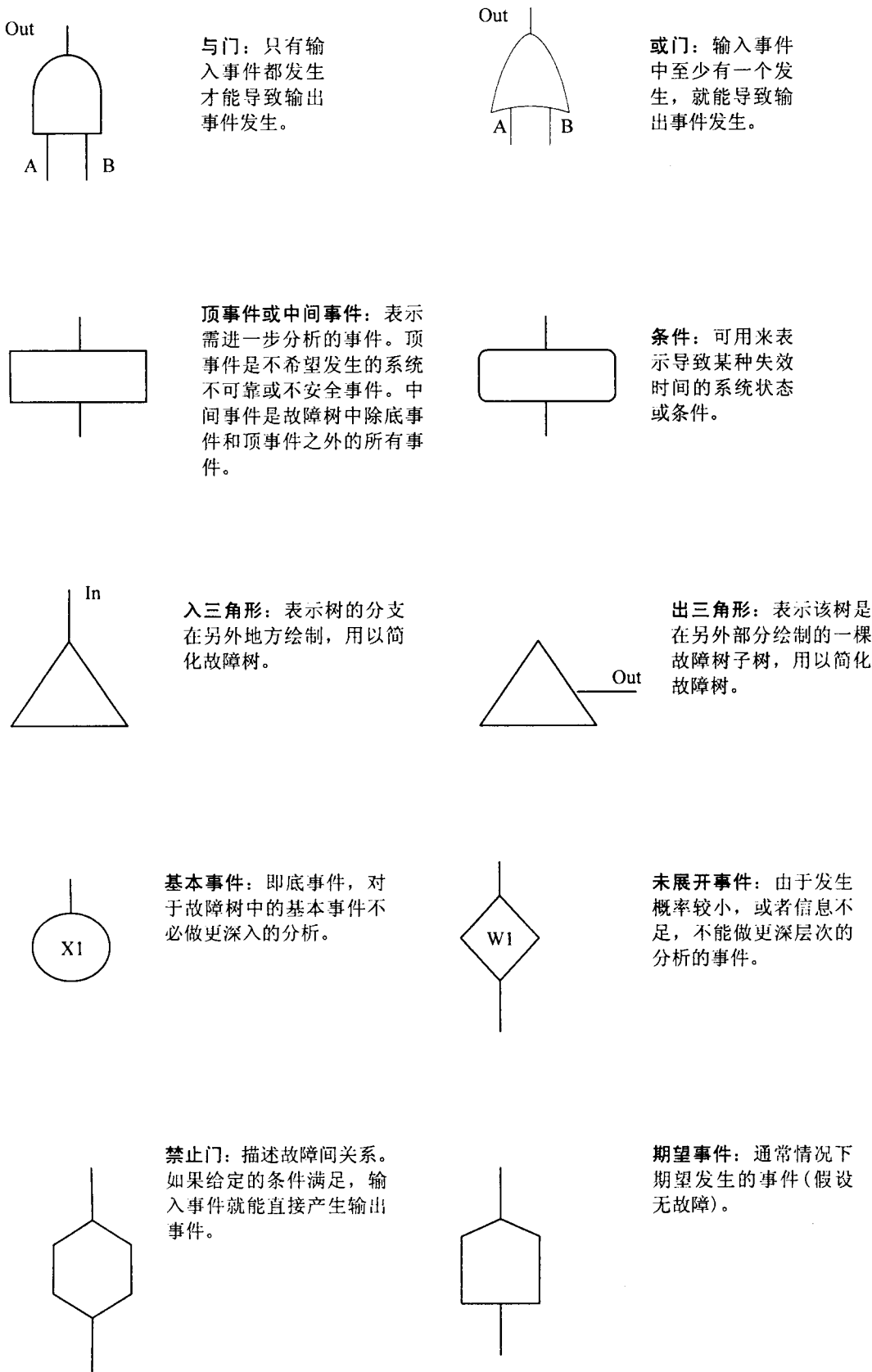


图 C.1 通用的软件故障树符号

C.2.2 软件故障树的定性分析

软件故障树的定性分析要求如下：

- a) 割集与最小割集：割集为能引起顶事件发生的基本事件集合，最小割集不包含任何冗余因素的割集。如果去掉最小割集中的任何事件或条件，它就不再成为割集。如果割集包含了多个相同的基本事件，则可以去掉冗余信息进行分析；如果一个割集是另一个割集的子集，那么在以后的分析中可以不考虑后者，因其并非最小割集。
- b) 最小割集的定性比较：当软件故障树建立并得出所有的最小割集后，应对其进行定性比较，并将结果应用于指导故障诊断，并指出改进系统的方向。最小割集的定性比较应遵循的原则为：首先根据最小割集所包含的底事件数目(阶数)排序，在各底事件发生概率比较小，其差别不大的条件下，比较按以下原则进行：
 - 1) 阶数越小的最小割集越重要；
 - 2) 低阶最小割集所包含的底事件比高阶最小割集的底事件重要；
 - 3) 在不同最小割集中重复出现次数越多的底事件越重要。

C.3 软件故障树分析(SFTA)应用示例

C.3.1 概述

检测中断屏蔽寄存器(IMR)软件是一个实时监控系统中对 8259 中断控制器进行检测的软件。

导致 8259 中断控制器检测失效的原因包括两个方面的内容：一是中断屏蔽寄存器(IMR)能否正确读写，二是外部中断能否被正确屏蔽。

检测中断屏蔽寄存器(IMR)软件是对中断屏蔽寄存器(IMR)进行读写操作，然后比较读写结果，以判明错误与否。由于实时监控系统的要求，检测不能破坏 IMR 原先的值，检测中断屏蔽功能实现屏蔽所有外部中断，再设置定时器通道 0，令其发出中断请求，最后检测系统是否响应中断。自检程序中先令定时器通道 0 每隔 $10\mu\text{s}$ 发一次中断， $20\mu\text{s}$ 后检测系统是否响应定时器中断。中断服务程序的作用是设置中断响应标志。正常情况下系统不响应定时器中断请求，否则中断控制器失效。8259 中断控制器检测流程框图如图 C.2 所示。

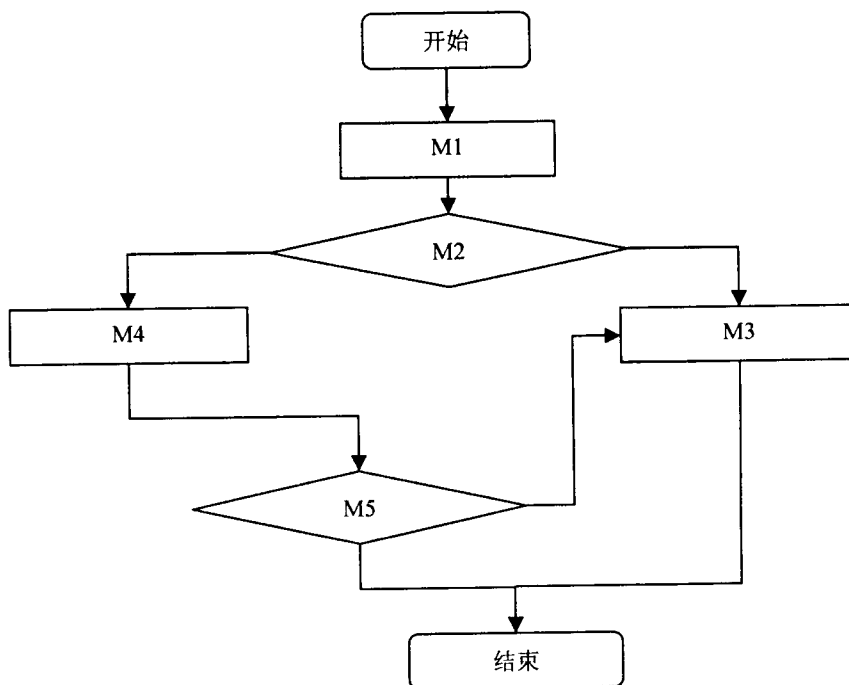


图 C.2 控制流图

C.3.2 SFTA 过程

SFTA 大致分为如下步骤：

- a) 分析对象定义：分析对象包括从入口至出口之间的程序及一个中断服务程序。
- b) 分析层次定义：分析的层次为软件单元，包括 5 个软件单元。
 - 1) M1：IMR 读写功能检测；
 - 2) M2：判定 M1 输出结果；
 - 3) M3：设置失效结果标志；
 - 4) M4：中断屏蔽功能检测(包括中断服务子程序)；
 - 5) M5：判定 M4 输出结果。
- c) 单元的控制流程见图 C.2。
- d) 假设条件：
 - 1) 各单元件信息(命令、数据)传递正确；
 - 2) 各单元间失效独立；
 - 3) 系统输入(启动)正确；
 - 4) 单元见实效不相互抵消。
- e) 事件定义：
 - 1) T：顶事件，中断控制器失效而未正确报告；
 - 2) T1：中间事件，IMR 读写功能失效而未正确报告；
 - 3) T2：中间事件，中断屏蔽功能失效而未正确报告；
 - 4) F1：底事件，读写地址失效；
 - 5) F2：底事件，读写结果判定失效；
 - 6) F3：底事件，失效结果标志设置失效；
 - 7) F4：底事件，中断屏蔽设置失效；
 - 8) F5：底事件，判定中断响应标志失效。
- f) 故障树建立：根据图 C.2，假设条件及事件定义，可得中断控制器的故障树，如图 C.3 所示。

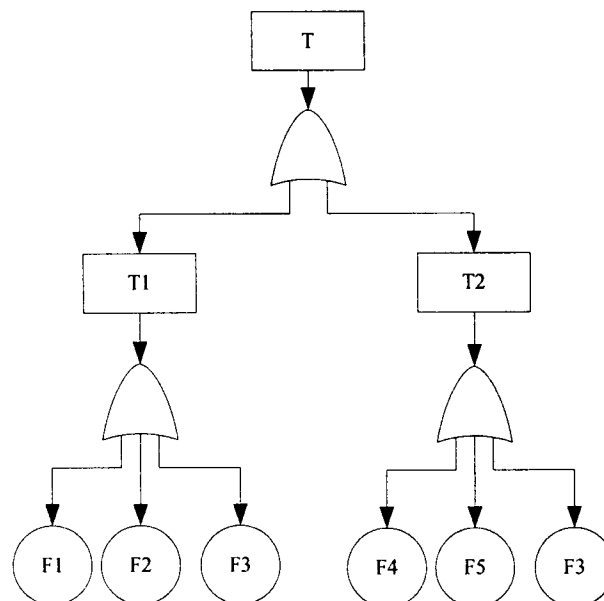


图 C.3 中断控制器检测程序的故障树

g) 定性分析:

$$T1=F1+F2+F3$$

$$T2=F4+F5+F3$$

$$T=T1+T2=F1+F2+F3+F4+F5$$

为了确保中断控制器检测功能正常完成,就要通过设计、审查、测试等手段保证 F1、F2、F3、F4、F5 不能出现。

附 录 D
(资料性附录)
X86 汇编语言安全性编码原则

D.1 强制类

D.1.1 模块中必须对使用的寄存器进行保护，在使用之前将它们的状态保存下来，使用结束后再恢复其内容。采用如下方式：

```

EXAMP      PROC  NEAR
            PUSH  AX
            PUSH  BX
            .....
            ; 其间用到寄存器AX和BX
            .....
            POP   BX
            POP   AX
            RET
EXAMP      ENDP
  
```

D.1.2 模块的 PROC 与 ENDP 配对使用。采用如下方式：

```

NAME      PROC
.....
NAME      ENDP
  
```

D.1.3 模块应是单入口、单出口结构。禁止如下方式：

```

MULTI_EXIT  PROC  NEAR
            .....
            RET
            .....
            RET
MULTI_EXIT  ENDP
  
```

D.1.4 模块禁止递归调用。禁止如下方式：

```

RECURSION  PROC  NEAR
            .....
            CALL  NEAR PTR  RECURSION
            .....
RECURSION  ENDP
  
```

D.1.5 模块或中断服务模块应以返回语句返回。采用如下方式(模块格式)：

```

EXAMP      PROC  NEAR
      .....
      RET
EXAMP      ENDP

```

采用如下方式(中断服务模块格式):

```

EXAMP      PROC  NEAR
      .....
      IRET
EXAMP      ENDP

```

D.1.6 压栈、出栈必须配对使用。采用如下方式(80X86):

```

EXAMP86      PROC  NEAR
      PUSH  AX
      PUSH  BX
      .....
      .....
      POP  BX
      POP  AX
      RET
EXAMP86      ENDP

```

采用如下方式(80X87):

```

EXAMP87      PROC  NEAR
      FLD   _      G_A1
      FMUL  _      G_B1
      FLD   _      G_A2
      FMUL  _      G_B2
      FADD
      FLD   _      G_A3
      FMUL  _      G_B3
      FADD
      FSTP  _      G_A
      RET
EXAMP87      ENDP

```

D.1.7 对未使用的中断源应绑定空函数。采用如下方式:

```

      .....                                ; 初始化使用的中断
MOV    AX, OFFSET C_FREE                  ; 初始化未使用的中断
MOV    WORD PTR[DI+28], AX
MOV    WORD PTR[DI+30], 0040H
      .....
C_FREE PROC
      IRET
C_FREE ENDP

```

D. 1. 8 在改变中断向量前应进行关中断的处理。采用如下方式：

```

CLI
.....
MOV     BX,      OFFSET      GINT    ; 中断服务模块入口地址
.....
STI

```

D. 1. 9 中断服务程序中应对浮点运算的浮点寄存器状态进行保护。采用如下方式：

```

FSAVE                                GMEMORY
.....
FRSTOR                               GMEMORY

```

D. 1. 10 中断内外若对同一变量赋值时，应考虑资源冲突。

D. 1. 11 整型有符号数比较后应使用有符号转移控制指令。采用如下方式：

```

G_A      DW      -1
.....
; 其间对 G_A 赋值操作
MOV      BX, G_A_
CMP      BX, 0
JLE_     A1
MOV      G_B1, 1
JMP      A2
A1:
MOV      G_B1, 2
A2:
.....

```

D. 1. 12 禁止使用段间跳转。

D. 1. 13 禁止使用跳转语句进行模块调用。禁止如下方式：

```

      EXAMP1    PROC    NEAR
.....
      JUMP     BEGIN
.....
EXAMP1    ENDP
BEGIN    PROC     NEAR
.....
BEGIN    ENDP

```

D. 1. 14 禁止条件判别成立时无转移控制执行语句。禁止如下方式：

```

.....
CMP      AX,    0
MOV      AX,    0
.....

```

D. 1. 15 变量使用前应赋初值，浮点数据要写小数点。采用如下方式：

```
DATA
NUM DD 1.0
.....
FLD NUM
```

D. 1. 16 禁止对有符号类型数据进行逻辑移位运算。禁止如下方式：

```
MOV G_NUM, -10
MOV BX, G_NUM
SHR BX, 2
```

D. 1. 17 禁止对有符号类型数据进行位运算。禁止如下方式：

```
MOV G_NUM, -10
MOV AX, G_NUM
AND AX, 0FCH
```

D. 1. 18 变量的使用禁止超出所定义数据类型的范围。禁止如下方式：

```
MOV AX, 8000000
```

D. 1. 19 禁止改变常量的值。禁止如下方式：

```
DATA DW 10 ; DATA为常数区数据
.....
MOV DATA, 20
```

D. 1. 20 禁止除数为零。

D. 1. 21 禁止数组越界。禁止如下方式：

```
PERSON_DATA STRUC
                ARG DW 6DUP(0)
                .....
PERSON_DATA ENDS
ERROR PROC NEAR
                .....
                MOV BX, 10
                MOV ARG[BX], AX
                .....
ERROR ENDP
```

D. 1. 22 开偶次方数应为非负数。禁止如下方式：

FLD	DATA1
FSUB	DATA2; 当 DATA1<DATA2时, 就会出错。
FSQRT	
FSTP	ST

D. 1. 23 进行反三角函数运算时, 应判断输入数据的合法性。

D. 1. 24 禁止使用浮点数作为循环判断变量。

D. 1. 25 浮点数不能作直接相等或不等的比较, 浮点比较后, 应使用 SAHF 取得比较状态再使用无符号转移控制指令。禁止如下方式:

FLD	QWORD	PTR	NUM1
FCOMP	QWORD	PTR	NUM2
JE	NEXT		

禁止如下方式:

FLD	QWORD	PTR	NUM1
FCOMP	QWORD	PTR	NUM2
FSTSW	AX		
SAHF			
JB	NEXT		

D. 1. 26 禁止宏嵌套定义。

D. 1. 27 宏定义形参之间应用逗号分隔。

D. 1. 28 禁止在模块中定义宏。

D. 1. 29 宏定义与宏引用的形参与实参应匹配。

D. 1. 30 禁止使用终止宏展开和取消宏定义的伪指令。

D. 1. 31 8087 指令后接着是 8086 指令, 两指令之间应加 `fwait` 等待指令。采用如下方式:

FSTSW	STAT
FWAIT	
MOV	AH, BYTE PTR STAT+1

D. 1. 32 在进行串处理前应使用 CLD 或 STD 明确串处理方向。采用如下方式:

MOV	SI, OFFSET	G_A1
MOV	DI, OFFSET	G_B1
MOV	CX, 10	
CLD		
REP	MOVSW	

D. 1. 33 对位进行判别时, 应使用 AND 或 TEST 指令。采用如下方式:

MOV	AX, CHECK_DATA
TEST	AX, 01H
JZ	NEXT
.....	
MOV	AX, CHECK_DATA
AND	AX, 01H
CMP	AX, 01H
JZ	NEXT

- D. 1. 34 程序区和数据区应分开定义。
- D. 1. 35 禁止存在多余物。
- D. 1. 36 禁止标识符使用关键字和保留字。禁止如下方式：

```
DATA
FLOAT DW 0
.....
PERSON_DATA      PROC      NEAR
                .....
                MOV      FLOAT, 1
                .....
PERSON_DATA      ENDP
```

D. 2 推荐类

- D. 2. 1 模块的圈复杂度不超过 10。
- D. 2. 2 模块扇出数不大于 7。
- D. 2. 3 模块调用深度不超过 6 层。
- D. 2. 4 单个模块的行数不超过 200 行。
- D. 2. 5 宏代码行数尽量少于 50 行。
- D. 2. 6 进行比较运算时，不宜使用相等比较。采用如下方式：

```
DEC    NUM
CMP    NUM, 0
JAE    NEXT
```

- D. 2. 7 不宜使用寄存器间接寻址跳转。
- D. 2. 8 不宜使用无限循环语句。
- D. 2. 9 不宜使用不同类型变量的混合运算。采用如下方式：

```
FLD    WORD      PTR      NUM1
FLD    DWORD     PTR      NUM2
FADD
FSTP   ST
```

- D. 2. 10 不宜使用 LOOP 循环嵌套。
- D. 2. 11 不宜使用 FINIT 语句。
- D. 2. 12 不宜使用 HLT 指令。
- D. 2. 13 不宜使用中断嵌套。
- D. 2. 14 在中断服务程序中不宜进行循环查询。
- D. 2. 15 有效注释应大于其代码行数的 20%。
- D. 2. 16 模块头部应加注释，内容应包括：模块名称、模块功能、模块的输入参数的名称和类型、量纲、模块的输出参数的名称和类型、量纲。
- D. 2. 17 建议标识符不包含小写字母“l”或大写字母“O”。

附 录 E

(资料性附录)

51 系列汇编语言安全性编码原则

E.1 强制类

E.1.1 应在中断服务程序入口处对 PSW 寄存器及使用的工作寄存器、专用寄存器、累加器进行保护，在执行 RETI 指令前恢复。采用如下方式：

```

TINT:
    PUSH    PSW
    PUSH    A
    PUSH    B
    .....
           ; 其间用到寄存器A和B
    POP     B
    POP     A
    POP     PSW
    RETI

```

E.1.2 对于不使用的中断应将中断允许寄存器 IE 相应位清零。采用如下方式：

```

MAIN:
    .....
    MOV     IE, #90H           ; 仅允许串行口中断，屏蔽其他中断。
    .....
    AND     IE, #EFH          ; 此处串行口中断不再使用，应关闭中断。

```

E.1.3 串行中断响应后，应在中断服务程序内对接收中断标志 RI、发送中断标志 TI 清零。采用如下方式：

a)	清除TI		
	MOV	IE, #90H	; 仅允许串行口中断
		
	ORG	0023H	
	LJMP	COMA	
		
	COMA:		; 串行口中断服务子程序
	CLR	ES	; 禁止串行口中断
		; 数据发送
	CLR	TI	; 清除发送中断标志
	SETB	ES	; 允许串行口中断
	RETI		
b)	清除RI		
	MOV	IE, #90H	; 仅允许串行口中断
		
	ORG	0023H	
	LJMP	SINT	
		
	SINT:		; 串行口中断服务子程序
	CLR	ES	; 禁止串行口中断
		; 取缓存区数据
	CLR	RI	; 清除接收中断标志
	SETB	ES	; 允许串行口中断
	RETI		

E. 1.4 定时器 2 中断响应后，应在中断服务程序内对定时器 2 溢出中断标志 TF2、T2EX 端出现负跳变中断标志 EXF2 清零。采用如下方式：

	ORG	002BH	; 定时器2溢出或T2EX端负跳变
	LJMP	INTR	; 定时器2中断
		
	INTR:		
		
	CLR	TF2	; 清除定时器2溢出中断标志
	(或CLR	EXF2)	; 清除T2EX端出现负跳变中断标志
	RETI		

E. 1.5 中断入口地址应使用 ORG 语句显示说明。

E. 1.6 未使用中断应采用空中断处理。采用如下方式：

	ORG	0003H	; 未使用中断，采用空中断
	RETI		

E. 1.7 子程序禁止递归调用。禁止以下方式：

```

RECU:
    .....
    ACALL    RECU
    .....
    RET

```

E. 1.8 子程序应为单入口、单出口结构。采用如下方式：

```

PROCNAME:
    .....
    JC      T1
    .....
    JB      T1
    .....
    .....
T1:      RET

```

```

PROCNAME:
    .....
    JC      T1
    .....
    JB      T2
T1:      RET
    .....
T2: RET

```

E. 1.9 禁止使用跳转语句进行子程序调用。禁止以下方式：

```

BEGIN:
    .....
    SJMP    SUMP
    RET
SUMP:      ; 子程序
    .....
    RET

```

E. 1.10 程序结束时，应使用 END 指令。**E. 1.11 子程序、中断服务程序应以返回语句返回。****E. 1.12 使用 ADDC(SUBB) 指令时，若不需加、减借位运算，应在执行 ADDC(SUBB) 指令前确保进位标志 C 为零。****E. 1.13 转移指令后，不应使用绝对地址。采用如下方式：**

```
LJMP    LABEL1          ; 此处使用标号
.....
ORG     4000H
LJMP    LABEL2
LABEL1:
MOV     R0,    #42H
.....
LABEL2:
```

禁止以下方式:

```
LJMP    4003H          ; 此处使用绝对地址
.....
ORG     4000H
LJMP    LABEL2
      MOV    R0,    #42H      ; 4003H地址处
.....
LABEL2:
```

- E. 1. 14 程序区和数据区应分开定义。
- E. 1. 15 应按照先入后出的原则进行压栈、出栈操作。
- E. 1. 16 程序中不应存在多余物。
- E. 1. 17 禁止使用关键字、保留字作标识符。
- E. 1. 18 变量初次被引用前应赋初值。
- E. 1. 19 禁止数组越界。禁止以下方式:

```
INIADDRESS    EQU    4020H
CONADD9852    EQU    4025H
.....
ORG    INIADDRESS
DB     00H, 04H, 1CH, 00H, 00H, 02H, 00H, 00H, 07H, 00H, 06H, 95H, 00H,
      00H
; 超出5个字节
```

- E. 1. 20 变量的使用禁止超出所定义数据类型的范围。
- E. 1. 21 禁止除数为零。
- E. 1. 22 建议常数以符号常量的形式表示。采用如下方式:

```
CK1        EQU          #31H
.....
MOV                A,          #CK1
.....
```

- E. 1. 23 有效注释应不少于代码的 20%。
- E. 1. 24 子程序头部应加注释。注释应包括子程序名称、实现的功能、输入参数、输出参数等。

E.2 推荐类

- E.2.1 子程序的圈复杂度不超过 10。
- E.2.2 子程序的扇出数不大于 7。
- E.2.3 子程序嵌套层深度不超过 5 层。
- E.2.4 子程序程序总行不超过 200 行。
- E.2.5 使用寄存器 R0~R7 前要设置 RS1、RS0，以确定寄存器工作区。
- E.2.6 推荐使用直接寻址，谨慎使用间接寻址。
- E.2.7 对于使用的中断，应在中断处理入口处使用跳转指令跳转到中断处理程序。采用如下方式：

```

ORG    000BH          ; 需要用到的中断
LJMP    SUB1
.....
SUB1:

```

- E.2.8 谨慎使用中断嵌套，使用时应进行状态保护。
- E.2.9 谨慎使用循环控制语句，避免出现死循环，中断服务程序应避免出现超时。
- E.2.10 在中断内外对同一变量操作时，应考虑操作冲突。
- E.2.11 在子程序入口处对 PSW 寄存器及使用的工作寄存器、专用寄存器、累加器进行保护，在执行 RET 指令前恢复。采用如下方式：

```

LCALL    SUMP
.....
SUMP:
PUSH     PSW
.....
POP      PSW
RET

```

- E.2.12 栈指针 SP 设为 1FH 或更大值。
- E.2.13 使用 ORG 应避免出现地址越界。
- E.2.14 变量名、标号名等标识符中谨慎使用小写字母“l”或大写字母“O”。
- E.2.15 保留字、标号、子程序名应统一大小写。

附 录 F
(资料性附录)
中断分析检查表

F.1 中断一般情况检查表

对中断的基本信息进行搜索，形成如表 F.1 所示的中断一般情况检查表。

表 F.1 中断一般情况检查表

序号	中断名称和标识	中断号	入口地址和标识	执行的功能	时间特性			优先级	追踪关系	备注
					周期/随机	频繁/偶发	执行时间			
1										
2										

F.2 中断使用情况一览表

在对中断使用情况进行了解的基础上，对每一个中断进行分析，形成如表 F.2 和表 F.3 所示的中断处理情况表和资源冲突检查表。

表 F.2 中断处理情况表

中断名称和标识		中断处理				
被分析项名称	子项名称	分析的判断			追踪关系	备注
		合格	疑问	不适用		
初始化	与应用要求的符合性					
	与硬件电路设计的符合性					
	与芯片使用手册的符合性					
现场保护和恢复	入口是否保护了所有用到的资源					
	出口处是否正确进行了恢复					
特殊情况处理	被屏蔽的处理					
	被推迟的处理					
	被遗漏的处理					
	有嵌套的处理					
异常情况	是否有自嵌套中断					
	是否有中断死锁					
开关中断时机	是否及时开中断					
	是否及时关中断					

表 F.2 (续)

中断名称和标识		中断处理				
被分析项名称	子项名称	分析的判断			追踪关系	备注
		合格	疑问	不适用		
定时中断						
中断响应时间	正常情况下是否满足要求					
	最坏情况下是否满足要求					

表 F.3 资源冲突检查表

端口/变量/ 缓冲器名称 和标识	主程序中访问			中断服务程序 m 中访问			……	中断服务程序 n 中访问			有无 冲突存在	追踪关系
	只读	只写	读写	只读	只写	读写	……	只读	只写	读写		
							……					
							……					

F.3 未使用中断检查

对未使用中断进行检查，形成如表 F.4 所示的未使用中断检查表。

表 F.4 未使用中断检查表

中断号	中断矢量表处理			中断控制芯片寄存器处理			是否显式关闭 该中断	追踪关系
	无处理	跳转到开始	跳转到陷阱	控制寄存器	状态寄存器	共享寄存器		

参考文献

GJB 2255 军用软件产品

中 华 人 民 共 和 国
国家军用标准
军用软件安全性设计指南
GJB/Z 102A—2012

*

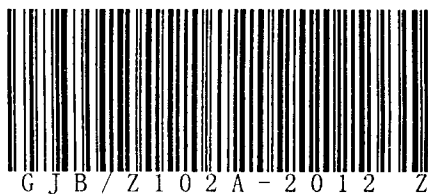
总装备部军标出版发行部出版
(北京东外京顺路7号)
总装备部军标出版发行部印刷车间印刷
总装备部军标出版发行部发行
版权专有 不得翻印

*

开本 880×1230 1/16 印张 3¼ 字数 101 千字
2012 年 10 月第 1 版 2012 年 10 月第 1 次印刷
印数 1—1000

*

军标出字第 8880 号 定价 49.00 元



G J B / Z 1 0 2 A - 2 0 1 2 Z