ICS 29.130.20
CCS K 30

中华人民共和国国家标准化指导性技术文件

GB/Z 41912—2022/IEC TR 63201:2019

低压开关设备和控制设备 嵌入式软件开发指南

Low-voltage switchgear and controlgear—Guidance for the development of embedded software

（IEC TR 63201:2019，IDT）

2022-10-12 发布　　2023-05-01 实施

国家市场监督管理总局
国家标准化管理委员会　发布

目　　次

前　言

本文件按照GB/T 1.1—2020《标准化工作导则　第1部分:标准化文件的结构和起草规则》的规定起草。

本文件等同采用IEC TR 63201 :2019《低压开关设备和控制设备嵌入式软件开发指南》。文件类型由IEC的技术报告调整为我国的国家标准化指导性技术文件。

本文件做了最小限度的编辑性改动:

——用GB/T 8566—2007代替了资料性引用的IEC 12207:2008;

——用GB/T 20438(所有部分)代替了资料性引用的IEC 61508(所有部分);

——用GB/T 21109.1—2007代替了资料性引用的IEC 61511-1:2016;

——用GB 28526代替了资料性引用的IEC 62061;

——用GB/T 34924—2017代替了资料性引用的IEC Guide 116:2018。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由中国电器工业协会提出。

本文件由全国低压电器标准化技术委员会(SAC/TC 189)归口。

本文件起草单位:上海电器科学研究院、上海正泰智能科技有限公司、常熟开关制造有限公司(原常熟开关厂)、浙江天正电气股份有限公司、施耐德电气(中国)有限公司上海分公司、加西亚电子电器股份有限公司、胜利油田恒源电气有限责任公司、江苏米特物联网科技有限公司、浙江聚创智能科技有限公司、上海电器科学研究所(集团)有限公司。

本文件主要起草人:黄兢业、郑捷欣、汪利敏、奚慎云、陶晓东、双兵、张新雨、管红武、史蒙云、吴桂初、薛吉、王军。

引　言

目前，越来越多的可编程电子产品被集成在开关设备和控制设备中。例如，软起动器、电子式过载继电器、带电子脱扣单元的断路器、内置微控制器的接近开关和一些附件，如扩展模块和控制面板等，正在使用带嵌入式软件（一般称为固件）的可编程电子产品。这种嵌入式软件通常支持设备提供的主功能，如过流保护和其他重要功能（例如监控设备的报警检测）。

与纯粹的机电设备相比，在开关设备和控制设备中集成嵌入式软件不应降低其主功能的完整性。因此，本文件提供了嵌入式软件的最低标准要求。

本文件参考了开发嵌入式软件自动化安全功能的现有最佳实践标准 GB/T 20438.3—2017。功能安全方法主要应用于机械、汽车、自动化和过程自动化，这些领域的安全功能是由多个组件实现的，这些组件在组合时应匹配一致并确保完整性水平。在其他领域，如配电和电力控制系统，过电流脱扣、剩余电流脱扣、负载监测等关键功能应遵循与系统安全性和可靠性相关的安装规定和协调规则。因此，本文件可被视为依据 GB/T 20438.3—2017 给出的良好实践。

本文件还将提供关于 UL 489:2016 的附件 SE 的最新方法。

本文件旨在提供以下方面的指导：

——与嵌入式软件有关的风险评估方面；

——嵌入式软件评价方法；

——软件架构；

——基本编码规则；

——控制软件出错的措施；

——软件验证及其与设备或系统验证的关系。

在本文件中，“软件”一词被用作嵌入式软件的广义术语。

低压开关设备和控制设备
嵌入式软件开发指南

1 范围

本文件提出了嵌入式软件的相关信息与推荐的最低要求，这些嵌入式软件支撑了开关设备和控制设备在全生命周期内主功能的实现。本文件还包括参数化要求和安全编码标准的基础要求。

如果产品标准中未涵盖本文件内容，则本文件可以作为产品标准的补充要求。

本文件适用于新产品开发或现有产品的重大改进。

本文件不包括 GB 28526、GB/T 16855.1 及 GB/T 20438(所有部分)中的机械或自动化控制系统的功能安全性要求，也不包括 ISO 27005 和 IEC 62443(所有部分)中的网络安全风险要求。本文件仅给出了安全编码规则的一些示例。

注：IEC TS 63208:2020 已发布，其基于 ISO 27005 和 IEC 62443(所有部分)规定了开关设备和控制设备中的网络安全措施。

2 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅该日期对应的版本适用于本文件；不注日期的引用文件，其最新版本(包括所有的修改单)适用于本文件。

GB/T 5271.1—2000 信息技术 词汇 第 1 部分：基本术语(eqvISO/IEC 2382-1:1993)

注：GB/T 5271.1—2000 被引用的内容与 ISO/IEC 2382-1:1993 被引用的内容没有技术上的差异。

3 术语和定义

GB/T 5271.1—2000 界定的以及下列术语和定义适用于本文件。

IEC 和 ISO 的术语数据库可以通过下述网址访问：

——IEC：http://www.electropedia.org/

——ISO：http://www.iso.org/obp

3.1

嵌入式软件 embedded software

由制造商提供的软件，是产品的组成部分，可以进行部分修改。

注 1：固件和系统软件都是嵌入式软件的示例。

注 2：嵌入式软件可以通过整体或部分更新来升级。

3.2

可编程电子 programmable electronic

以计算机技术为基础，一般由硬件、软件及其输入和(或)输出单元构成。

示例：下列均是可编程电子装置：

——微处理器；

——微控制器；

——可编程控制器；

——专用数字集成电路(带有可编程部分的 ASICs)；

——可编程逻辑控制器(PLCs)；

——其他以计算机为基础的装置(例如智能传感器、智能变送器、智能执行器)。

注 1：本术语包含基于一个或多个中央处理器(CPUs)及相关的存储器等为基础的微电子装置。

注 2：术语“可编程组件”来源于 ANSI/UL 1998:2013 中定义 2.39。其定义为“可在设计中心、工厂或现场编程的任何微电子硬件。其中，术语“可编程”是指“可通过操作软件来改变组件行为的方式”。

[来源：GB/T 20438.4—2017，3.2.12，有修改]

3.3

主功能　main function

(开关设备和控制设备中)定义的功能，其故障可导致非预期的操作，从而导致危险情况、失去保护功能或失去制造商规定的关键功能。

3.4

系统性失效　systematic failure

因某些确定的原因引起的失效，只有对设计或制造过程、操作规程、文档或其他相关因素进行修改后，才有可能消除这种失效。

注 1：未经修改而仅进行修正性维护，通常无法消除失效原因。

注 2：通过模拟失效原因可以诱发系统性失效。

注 3：系统性失效的原因可包括以下情况中的人为错误：

——安全要求规范；

——硬件的设计、制造、安装和(或)运行；

——软件的设计和(或)实现。

[来源：GB/T 20438.4—2017，3.6.6，有修改]

3.5

全可变语言　full variability language；FVL

可提供实现各种各样功能和应用的能力的一种语言。

注 1：嵌入式软件通常使用 FVL，应用软件则很少使用 FVL。

注 2：FVL 实例包括：Ada、C、Pascal、指令表、汇编语言、C++、Java 和 SQL。

[来源：GB/T 21109.1—2007，3.2.81.1.3，有修改]

3.6

配置管理　configuration management

系统演变过程中其组件的标识规则，用以控制这些组件的变更并保持在生命周期全过程中的连续性和可追溯性。

[来源：GB/T 20438.4—2017，3.7.3]

3.7

基线　baseline

在特定时间点商定的一套设备要素(硬件、软件、文档、测试……)的基准，作为验证、确认、修改和变更的基础。

注：如果一个要素被变更，则在新基线被定义前，其基线状态一直处于中间态。

3.8

编码规则　coding rules/coding standard

为确保程序的可读性、可维护性、兼容性和鲁棒性而对软件源代码进行格式化规定的一套规则和指南。

注：编码规则的典型应用为命名约定、文件命名和分配、格式和缩进、注释和文档、类、函数和接口、允许/禁止的标

准库函数用法、数据类型、指针和引用用法以及测试等。

3.9

软件单元　software unit

完成某个特定功能的最基本的程序段。

注：软件单元又称为软件模块和软件组件，如国际软件测试资格委员会(ISTQB)文件中的规定。

[来源：GB/T 8566—2007，3.29，有修改]

3.10

集成测试　integration tests

在验证和系统确认之前，在软件单元和硬件/软件集成过程中进行的(软件)测试，用来以验证软件和硬件的兼容性。

[来源：IEC 60880:2006，3.23，有修改]

3.11

软件验证　software verification

通过检查(如测试、分析)软件，确认其集成或单元功能满足需求。

3.12

静态分析　static analysis

可发现软件存在缺陷的源代码特征检查。

注1：静态分析通常会发现无法访问的代码段、未使用的、误用的、重复定义的或者未进行初始化的变量，以及意外的执行路径。

注2：静态分析通常运用计算机辅助软件工程工具。

[来源：GB/T 2900.99—2016，192-09-22，有修改]

3.13

系统验证　system validation

通过检查和提供客观证据确认满足设备或系统的特定预期应用的需求。

注：原则上，嵌入式软件集成在设备或系统中。该设备或系统的验证包括与嵌入式软件相关的验证。

4　主功能的风险评估和识别

根据制造商的经验，在设备预期应用的背景下进行的系统分析，包括其不同的操作模式以及合理可预见的误用，宜确定主功能清单及其相关风险等级。

例如：对象的传感器检测、过电流保护功能、供电连续性、电机系统断电。

为此，宜采用 GB/T 34924—2017 等风险评估方法。

实现主功能的每个软件部分宜按照本文件提供的方法进行管理。

5　设计管理

5.1　目标

需要由专门的软件设计组织(团队)来确保主功能能够完全按照其原始规范实现。

该组织宜在软件管理计划中规定。软件管理计划可能是全局设计管理计划中明确规定的一部分内容。

5.2　主功能的软件管理计划

本计划用于定义整个软件开发生命周期中的活动管理，以制定规范并验证与主功能(见 3.3)相关的

软件。

该组织宜定义管理(启动、控制)和活动执行的角色和相关责任。

该计划宜根据需要制定、记录以及修改,还要提供相应的措施防止出现不正确的需求规范定义、实现或变更问题。

主功能的软件管理计划宜与项目相适应。

特别是,该计划宜:

a) 确定开发主功能相关部分所需的设计活动(按适当顺序组织的必要设计活动如图 2 所示);

b) 描述满足主功能相关规定需求的政策和策略;

c) 描述开发、集成和可能包含合格评定的验证策略;

d) 确定负责执行和审查每个主功能设计活动的授权人员、部门或其他组织单位和资源。确定时宜考虑适当的能力水平(如培训程度、技术知识、经验和资质);

e) 确定或建立用于记录和维护设备主功能相关信息的程序和资源,并考虑风险评估结果和变更管理;

f) 描述配置管理策略,考虑相关的组织问题,如授权人员和组织的内部结构等;

g) 描述修改策略;

h) 建立软件验证计划(详见 7.4.3)。

5.3 配置管理

配置管理是一种系统工程管理过程,用以确保设备在整个生命周期内,其性能、功能、物理特性与需求、设计和运营信息中的规定一致。配置管理的内容包括了验证设备的软硬件要素是否已得到详尽的标识与记录,从而为设备在整个生命周期内提供支持。该过程有助于管理设备信息及设备变更,以具备修改的能力;可改善设备的性能、可靠性或可维护性;延长设备使用寿命;降低成本;降低风险和不利因素;或校正缺陷。

配置管理的主要内容包括:

——通过识别设备的各要素(如系统、子系统、功能、功能块、管理文件、基线创建工具),识别设备的结构;

——控制各生命周期阶段内某一特定时间点所创建的要素发布;

——记录并报告基线中的一部分和(或)即将成为基线一部分的要素状态;

——审查并复审所有要素,并保持基线与所有要素间的一致性。

在设备系统和软件整个开发周期内,宜为设备的配置管理制定相关程序,尤其是:

a) 在特定阶段执行正式配置控制的节点;

b) 用于唯一识别设备(硬件和软件)所有组成部分的程序;

c) 用于防止未经授权的要素投入使用的程序。

配置管理程序宜按照软件管理计划执行。

为确定每个设备版本(包括所有相关配置项)的唯一基线,宜在“变更—控制—处理”过程中考虑程序的要求。

5.4 变更管理

如要实施与主功能相关的修改,则宜明确相关活动。并在进行任何修改前编制、记录和批准该行动计划。

注 1:修改请求可能来自:

——主功能需求规范已变更;

——实际使用条件;

——事件/事故经验；

——加工材料的变更；

——设备或其操作模式的修改。

注 2：根据设备使用信息或说明手册对设备进行的干预(如调整、设置、修理)不视为本条款中的修改。

宜记录请求修改的原因。

宜对修改后的效果进行分析，以确定其对主功能的影响。

宜记录修改对设备主功能的影响及修改的效果。

对设备有影响的所有已接受的修改宜使其硬件和(或)其软件返回到适当的设计阶段(例如，规范、设计、集成、安装、调试、验证和系统验证)。后续所有阶段和管理程序宜按照本文件中针对特定阶段规定的程序执行。所有相关文件均宜相应校对、修订和重新发布。

5.5 缺陷管理

对于任何软件开发，都要求在整个开发周期内实行可靠的缺陷管理流程。

缺陷可能是：

——编码错误引起的结果；

——与原始需求的偏差；

——外部事件引发的非预期行为。

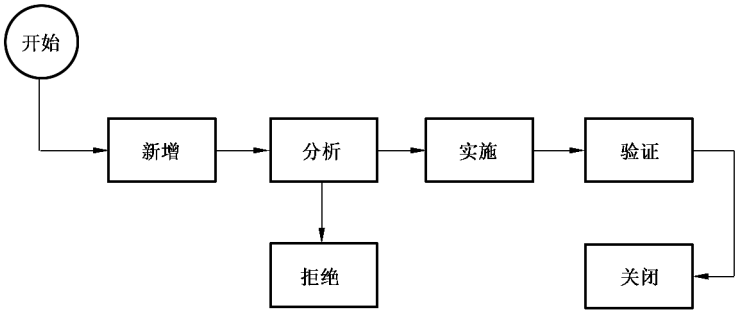

图 1 缺陷管理流程

缺陷管理工作流程因开发项目组的实践而异。图 1 给出了缺陷生命周期流程示例，包括如下阶段：

——新增：这是识别缺陷流程的第一步。通常由测试团队提交缺陷，有时客户也会报告缺陷。

——分析：一旦发现缺陷，开发团队尝试各种方法重现并寻找缺陷的根本原因。通常在此步骤中确定或修改缺陷的严重性。

——拒绝：根据缺陷类型或缺陷的严重程度，可拒绝解决缺陷。

——实施：开发人员修复了缺陷，并将其放置在原先缺陷发生处。

——验证：缺陷修复后，验证团队将验证缺陷是否已实际修复且系统是否按预期工作。

——关闭：一旦缺陷被修复、记录和解决后，其状态将被标记为关闭。

注：可以将缺陷管理工作流程的每个状态分配给特定的项目团队成员，并指定交付期限。

5.6 系统构建与发布流程

5.6.1 二进制代码生成

生成在设备(开关设备和控制设备)上运行的软件二进制文件是高度可预测的，即相同的源码应生成相同的二进制文件。

因此，生成工具(编译器、链接器、下载器……)及其使用的选项应受控于配置管理。

5.6.2 发布管理

发布软件时，宜使用唯一的版本号/修订号进行标识。最常用的编号习惯包括主版本号、次版本号和修订号：

——当软件兼容性变更或增加多个新功能时，主要版本号需要递增。通常，当主版本号为 0 时，该软件仍处于开发阶段或 Beta 版本。

——当增加少量的附加功能时，次版本号需要递增。

——当缺陷被修复时，修订号需要递增。

每个软件版本都应通过注释的方式进行记录，这些注释详细说明了发布内容：新功能、已修复缺陷列表以及已知的限制。

6 嵌入式软件的手动参数化

6.1 一般要求

有些设备需要通过手动参数化来实现主功能。可通过连接到设备的远程连接（如基于 PC 的配置工具）或人机界面[如显示器、指拨开关（DIP 开关）、电位计、旋钮、存储卡]，进行参数设定。

基于软件手动参数化要求的目的是确保正确选择与主功能相关的参数，并将其传输到执行主功能的设备中。这些参数不宜以意外或未经授权的方式影响主功能。

可以采用不同的方法来设置这些参数。甚至基于 DIP 开关的参数化都可用来设置或更改与主功能相关的参数。但是专用参数化的软件工具（通常称为配置或参数化工具）正变得越来越普遍。本条款仅适用于获得授权的人员执行或控制基于软件的手动参数化。如在设备的首次调试时或设备生命周期内的任一时刻需要设定参数，则本条款适用。

6.2 主功能相关参数的影响因素

在基于软件的手动参数化过程中，主功能参数可能会受下列因素影响，如：

——负责参数化人员的数据输入错误；

——参数化工具中软件自身故障；

——参数化工具随附的软件和（或）服务故障；

——参数化工具硬件故障；

——参数化工具向设备传输参数时出现故障；

——设备正确存储参数时发生故障；

——参数化过程中受到的系统性干扰，如电磁干扰或断电影响；

——受外部影响或其他因素引起的干扰，如电磁干扰或（随机性）断电。

如果没有适当的措施，上述影响可能会在不通知参数化负责人的情况下发生，并导致如下情况：

——参数化过程中无法完全或部分更新参数；

——参数完全或部分不正确；

——通过有线或无线网络传输参数时，将参数传输到错误的设备。

宜采取措施抵消、避免或控制由上述影响引起的潜在危险故障。

6.3 基于软件的手动参数化要求

设备或其相关子系统的供应商宜为基于软件的手动参数化提供专用工具。该工具宜带有标识（名称、版本等）。设备或其相关子系统和参数化工具宜能够防止未经授权的更改，如采用密码等。

当参数化过程可能会导致不安全状态时，宜在设备处于安全状态的情况下离线进行参数化。此外，

还宜满足以下要求：

a) 基于软件的手动参数化设计宜被视为设备设计中与主功能相关的设计，并在主功能需求规范中加以描述，如在系统需求规范中；

b) 设备或子系统宜提供数据检验系统，该系统具备诸如检查数据极值、格式和(或)逻辑输入值等功能；

c) 宜维护用于参数化的所有数据的完整性。为此，宜采取以下措施：

——控制有效输入范围；

——控制传输之前的数据损坏；

——控制参数传输错误带来的影响；

——控制参数传输不完全带来的影响；

——控制参数化的硬件和软件的故障和失败所带来的影响。

在传输/重传过程中用于编码/解码的软件单元和用于向用户显示主功能相关参数的软件单元至少宜使用功能多样性来避免系统性失效。在这种情况下，多样性包含使用完全不同的传输方法来降低发生常规失效的可能性。

6.4 参数化工具的验证

至少宜采用下列方法对参数化工具的基本功能进行验证：

——验证每一个主功能相关参数是否正确设置(检查有效值)；

——验证每一个主功能相关参数的合理性，如检测无效参数组合等；

——验证是否提供了防止未经授权修改主功能相关参数的措施。

注：当使用非专用设备(如个人计算机或类似设备)进行参数化时，这一点尤为重要。

6.5 基于软件手动参数化的文档管理

基于软件的手动参数化宜使用设备供应商提供的专用参数化工具进行，并宜根据使用信息中规定的要求进行记录。该记录信息可由各方创建。记录内容可以保存在不同介质(纸张、电子化、参数化工具中、设备上等)。宜激活并采用对未授权存取提供的保护措施。

宜记录初始参数化以及对参数化的后续修改。记录文档中宜包括：

a) 当文件不在内部存储时，需记录参数化设备的标识；

b) 初次参数化或更改参数的日期；

c) 数据设置的日期或版本号；

d) 执行参数化的授权人员的姓名；

e) 标识所用数据的来源(例如，预定义的参数集，保留最后一次更改后的前一次设置)；

f) 明确标识主功能相关参数。

7 设计生命周期

7.1 一般要求

嵌入式软件的所有生命周期活动宜着重于避免在其设计生命周期中引入错误，以下要求的主要目的是生成具有可读性、可理解性、可测试性、可维护性的正确软件。

如果软件同时执行非主功能和主功能，则所有软件都宜视为与主功能相关，除非在设计中可证明功能之间有足够的独立性。因此，在可行的情况下，宜将基本设备功能等非主功能与主功能分开。

7.2 工具选用

宜选用一套合适的工具，包括配置管理、仿真和带有测试发生器（例如“分线盒”）的试验设备。宜考虑在整个生命周期中是否有合适的工具以便于设备维护、升级与参数设置。宜在配置管理文件中解释并记录工具的适用性。

适用性的证明方法如下：

——选用恰当的避免和控制故障措施，进行严格的测试来验证其有效性，并记录结果；

——分析并确认这些工具的失效在工具链中可能带来的影响。

注 1：避免和控制故障措施的合理性取决于故障的严重程度。评估严重程度的基础是分析。只有在对支持工具和设备的应用有一定了解的情况下，才能进行该分析。

注 2：失效的影响可能因不同的支持工具而异。GB/T 20438.4—2017 将软件开发生命周期内使用的离线支持工具分为三类，如在软件生命周期之外使用，需要分析离线支持工具的分类是否合适。

注 3：关于支持工具的定义和示例，见 GB/T 20438.4—2017。

注 4：本文件未规定避免或控制离线支持工具故障的任何措施。有关示例见 GB/T 20438.3—2017 中的 7.4.4。

注 5：可采取适当措施确保制造过程中使用的软件工具（编程和配置工具）不会改变嵌入式软件的完整性。

7.3 软件生命周期

7.3.1 软件生命周期模型

宜使用分解为不同阶段的软件生命周期模型（例如 V-模型），包括管理和文档记录。

在满足本条款所有目标和要求的前提下，可以使用任何软件生命周期模型。嵌入式软件宜按照 7.13 进行验证。

嵌入式软件使用全可变语言。图 2 为具体的 V-模型。

注 1：在 V-模型的左侧，审查了每个阶段的输出。审查是指根据某一阶段输入的需求，检查 V-模型中该阶段的输出。箭头“审查”表示软件验证的第一步。

注 2：适合项目规模和范围的项目管理技术和流程是最恰当的方式。

注 3：在 V-模型中，反馈回路箭头“验证”表示符合规范的测试用例结果。此外，还表示需要更精确的测试用例要求和规范。

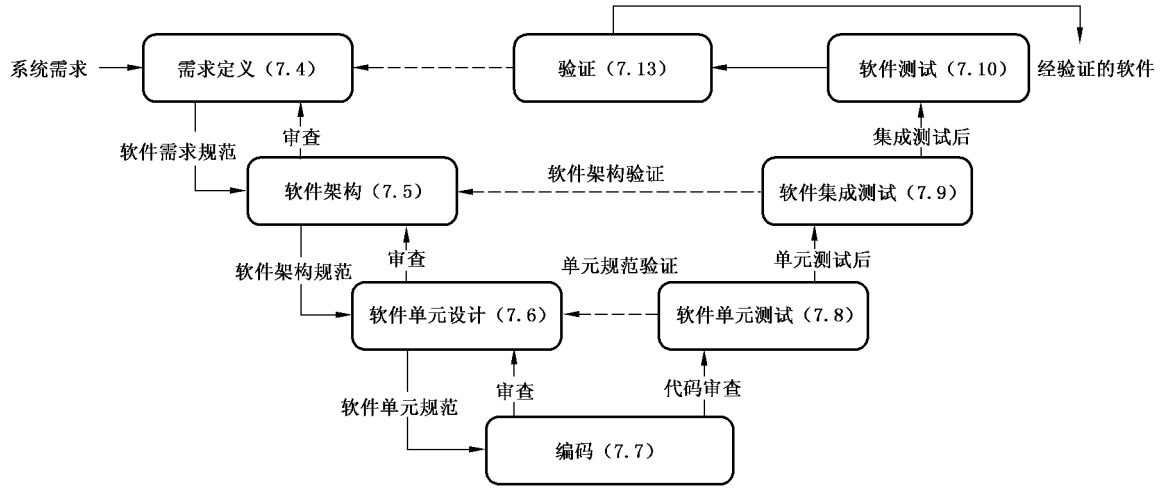

图 2 软件生命周期的 V-模型

7.3.2 审查、测试和验证的独立性

当本文件要求进行审查或测试/验证时，宜由不直接参与嵌入式软件设计的人员来执行，如独立于设计过程之外的人员。相关方可以是组织层级中不隶属于设计部门的其他人员、部门或组织。独立性等级宜与风险相对应。

独立性等级可以：

——其他人员；

——独立人员；

——独立部门；

——独立组织。

根据公司组织和公司内部的专长，有时可能需要外部组织来满足对独立人员和部门的要求。但如果公司拥有擅长风险评估和主功能应用的内部机构，且该内部机构独立于（通过管理和其他方法）主要开发人员，则这些公司可以使用自己的资源来满足独立组织的要求。

“其他人员”可以参与同一项目，但可能不参与相同的软件设计活动。“独立人员”可能参与同一项目，但不参与软件设计活动也不负责项目管理，同时也不能是上级领导。“独立部门”不能参与项目设计。“独立组织”则是在管理及其他资源方面独立于负责项目设计的组织。

当出现下列因素，独立部门比独立人员更合适：

——对类似设计更丰富的经验；

——项目具有较高复杂度；

——项目规模较大。

7.4 需求定义

7.4.1 一般要求

宜根据软件系统需求来开发软件，并在整个设备生命周期内对软件进行管理。

软件需求规范详见 7.4.3。

7.4.2 系统需求

为支持软件开发过程，有必要提供以下信息：

a) 主功能规范；

b) 设备的配置或架构（例如：硬件架构、接线图、与主功能相关的输入和输出）；

c) 响应时间的要求；

d) 操作员接口和控制，例如：开关、操纵杆、模式选择器、拨号盘、触控设备、键盘等；

e) 设备的相关操作模式；

f) 对硬件诊断的要求，包括传感器、终端执行器等的特性；

g) 机械公差的影响，例如传感器和（或）其传感计数器部件的影响。

7.4.3 软件需求规范

7.4.3.1 设计需求

软件需求规范宜：

——具有结构化、可审查性、可测试性、可理解性、可维护性、可操作性。

——根据设备规格和架构为每个子系统设计规范。

——足够详细，以便进行适当的软件验证和测试。

——可追溯到设备的系统需求规范。这意味着该规范是可以理解的，以便其他人（例如非软件专家）可以验证该规范是否符合风险评估中定义的软件相关系统需求。

——不含模棱两可的术语和不相关的描述。

软件需求规范的输入宜可以直接与预期的输出相关联，反之亦然。文档中宜使用易于理解的半形式化方法，例如因果表、逻辑描述、表或图、功能块或时序图。

软件需求规范中宜规定下列内容：

a) 主功能的逻辑，包括输入和输出以及对检测到故障的正确诊断。可能的方法包括但不限于因果表、书面说明或功能块。

注 1：硬件也可以检测到故障（如输入卡检测到的信号差异）。这种故障检测是软件需求规范的一部分。

b) 测试用例规范，包括：

——进行测试所需的特定输入值、环境参数、预期试验结果（包括合格/不合格标准）；

——故障插入或注入。

注 2：对于简单功能，可以通过主功能的规范连带给出测试用例。

c) 输入设备（如传感元件和开关）和最终控制元件（如电磁阀、继电器或接触器）的诊断功能。

d) 使设备达到或保持安全状态的功能。

e) 与故障检测、通知和处理相关的功能。

f) 与设备在线和离线周期性测试相关的功能。

g) 对设备控制流和数据流进行自我监控。故障检测时，宜采取适当的措施以达到或保持安全状态。

h) 防止未经授权修改设备的功能。

i) 与其他设备的接口。

j) 主功能响应时间。

k) 信号处理功能的算法模型。

l) 编码规则。

注 3：软件文档指南在 GB/T 20438（所有部分）和 ISO/IEC/IEEE 26512:2018 中给出。

宜根据系统需求规范审查软件需求规范中的信息，并在必要时进行修订，以确保充分规定软件需求。

7.4.3.2 编程语言规范及相关开发方法

在设计嵌入式软件时，如使用了全变量语言，下列附加要求适用。

当使用具有同等效力的替代技术和措施时，宜考虑 GB/T 20438.3—2017 中附录 A 和附录 B 的表格要求。

所选语言的设计和选择宜考虑以下特性，具体取决于应用程序的复杂性。

a) 抽象、模块化和控制复杂性的其他特性；在可能的情况下，软件宜基于经过充分验证的逻辑功能，其中可能包括用户库函数和用于链接逻辑功能的明确规则。

b) 表达方式：

——功能，理想情况下作为逻辑描述或算法功能；

——模块化元件之间的信息流；

——时序和时间相关要求；

——时间限制；

——对共享资源的并发和同步访问；

——数据结构及其属性，包括数据类型、数据范围的有效性；

——异常处理。

c) 易于开发人员及其他需要了解设计的人员理解的说明，包括对应用程序功能的理解和对设备技术限制的了解。
d) 软件验证计划，包含对主功能完整性的评价方法、验证策略与工具、对结果的评估以及如何采取纠正措施。
e) 系统验证计划的相关部分，宜包括预期的使用测试用例。
f) 维护步骤，如修正。

7.5 软件架构

7.5.1 一般要求

为实现软件需求规范中的要求，宜建立软件架构。软件架构定义了软件的主要组件和子系统，规定它们如何相互连接，以及如何实现所需的属性。软件架构还需定义软件的总体行为，以及软件组件如何交互及相互影响。主要软件组件的示例包括操作系统、数据库、输入/输出子系统、通信子系统、应用程序、编程和诊断工具等。

软件架构宜遵循模块化的方法，即在有限的软件单元大小内完整定义一个接口和用于子程序和函数的入口/出口。每个软件单元应有一个清晰易懂的任务，并且应限于一个完整的主功能。

7.5.2 软件架构规范

宜提供软件架构规范作为软件架构设计的输出，并用以解释主要的软件内容。例如下面列表中所示：

——软件架构，定义了为满足软件需求规范而规定的架构；
——全局数据；
——所用的数据库；
——所用的已有软件单元；
——诊断功能(内部、外部)；
——拥有唯一性标识信息的编程工具；
——软件集成测试用例和过程，包括测试环境规范、支持软件、配置描述和对测试失败后采取纠正措施的过程。

软件架构规范中包含的信息宜根据软件需求规范进行审查。

7.6 软件单元设计

7.6.1 一般要求

如果以前开发的软件单元要作为设计的一部分重新使用，宜分析其在满足主功能软件需求规范方面的适用性。还需评估来自先前软件开发环境的限制(例如操作系统和编译器依赖性)。

7.6.2 输入信息

对于软件单元，软件架构规范中宜规定下列信息：

——软件单元描述；
——软件单元接口(输入和输出及其数据类型)，以及(如需要)数据范围；
——所用软件单元库的标识；
——特殊编码规则。

7.6.3 软件单元规范

软件单元规范宜包含下列信息：

——每个软件单元的逻辑(如功能)描述；

——每个软件单元输入输出接口的完整定义；

——输入输出数据的格式和取值范围及其与软件单元的关系；

——包括正常和非正常操作的测试用例；

注：尽管测试用例通常包括在其指定范围内对参数进行单独测试，但是这些参数的不同组合可能会导致非预期的操作。

——中断的记录文件。

宜对照输入信息(见 7.6.2)审查上述信息。

7.7 编码

宜按照软件单元规范和编码规则开发软件。编码规则可以是众所周知的行业标准，也可以是制造商规定内部标准。宜按照软件单元规范和编码规则对代码进行审查。

注 1：编码规则旨在限制编程的自由度，以避免程序代码变得难以理解。

注 2：编码规则通常定义编程语言的子集或使用强类型编程语言(见 GB/T 20438.7—2017 的 C.4.1，C 语言在关键系统中的使用指南(MISRA C))。

注 3：代码审查可以通过手工进行，也可以通过静态代码分析工具自动进行。

宜使用以下编程手段来避免系统性失效：

——变量和配置参数的范围检查和合理性检查；

——时间和(或)逻辑程序时序监控，以检测有缺陷的程序时序；

——限制全局变量的数量或范围。

注 4：有关面向对象的体系架构和设计的指南，见 GB/T 20438.7—2017 中的附录 G。

编码输出物宜包含：

——源代码列表；

——代码审查报告。

为了限制网络安全风险，宜采用安全编码规则。根据网络安全需求、嵌入式操作系统和编程语言的不同，安全编码规则可能会有很大的不同。

例如，在高完整性、中等可用性、操作系统仅限于制造商应用程序和 C 语言的情况下，宜使用 ISO/IEC TS 17961:2013 和 CERT/CC C:2016 中规定的以下规则：

——从不信任用户输入：清洗传递给复杂子系统的数据，并从格式字符串中排除用户输入；

——使用容器类型时检查边界：不要形成或使用越界指针或数组下标，不要对指向非数组对象的指针进行整数的加减；

——安全内存管理：为对象分配足够的内存，且只动态分配空闲内存；

——仅在信号处理程序中调用安全函数：仅在信号处理程序中调用异步安全函数，并且不访问信号处理程序中的共享对象；

——其他规则：默认拒绝访问并遵守最小权限原则。

7.8 软件单元测试

未经评估的每个软件单元宜根据软件单元规范中定义的测试用例进行测试。

如果软件单元未通过测试，则宜采取预定义的纠正措施。

测试结果和纠正措施宜形成文件。

软件单元测试宜至少使用动态分析技术并进行测试(GB/T 20438.7—2017 中的 B.6.5)。

7.9 软件集成测试

宜根据软件架构规范中规定的集成测试用例进行测试。软件集成测试的结果宜形成文件。

宜执行以下两个不同的步骤:

a) 软件集成,仅检查软件模块的交互(通过专用软件集成测试工具);

b) 软硬件集成,验证上述相同的模块在目标硬件上的相互影响(允许有效的定时测量和全局行为检查)。

注:这些测试的目的是表明所有软件单元和软件组件/子系统都可以正确交互以执行其预期功能,并且不执行非预期功能。这并不意味着测试所有输入组合,也不意味着测试所有输出组合。测试所有等效类或基于结构的测试就足够了。边界值分析(GB/T 20438.7—2017 中的 C.5.4)或控制流分析(GB/T 20438.7—2017 中的 5.5.9)可以将测试用例减少到可接受的数量。可分析的程序更容易满足需求。

7.10 软件测试

7.10.1 一般要求

软件测试的主要目的是确保实现软件需求规范中的详述的功能。

软件测试的主要输出是一个文档,例如带有测试用例和测试结果的测试报告,且允许对测试覆盖率进行评估。

软件测试也包括故障仿真和相关的故障处理。

当使用已经测试过、包含故障检测和处理措施的软件单元时(例如输入信号的差异或输出的反馈节点),则不需要对这些故障检测和处理措施进行测试。在这种情况下,只需要对这些软件单元的集成进行测试。

如果依据指定最终用户用例(如开箱即用体验或软件升级)对集成在设备或系统中的目标硬件进行测试,则可以将软件测试作为系统验证的一部分进行。

宜采用功能测试作为基本措施。如可行,宜通过仿真来测试代码。

宜定义用于测试嵌入式软件的一般准则或程序,包括:

——测试类型;

——测试设备的规范,包括工具、支持软件和配置说明;

——测试及纠正嵌入式软件过程中的软件版本管理;

——缺陷跟踪过程,如错误登记、分级、修复、批准;

——测试失败后的纠正措施;

——完成相关功能或需求的测试标准;

——测试的物理位置,例如可以通过计算机仿真、工作台或设备硬件进行测试。

7.10.2 测试计划与执行

基于测试用例的测试计划宜包括:

——按名称定义角色和职责;

——安装测试环境;

——功能测试。

软件测试包括以下两种活动类型:

——静态分析:对软件的分析。例如通过人工审查,如“4 眼”代码审查:检查、走读、控制流分析、数据流分析。或通过自动代码分析,例如通过 MISRA 检查。静态分析通常由代码开发人员在代码实现过程中执行。

——动态测试:软件在受控且系统化的条件下执行,从而证明其达到预期功能,且无非预期的行为。尤其包括功能性测试或黑盒测试。

在软件生命周期的早期阶段,采用静态分析进行验证。当代码生成并可运行时,可以进行动态测试。为了验证软件生命周期活动的输出,上述两种类型的活动要结合使用。有关静态分析和动态测试的进一步说明,见 GB/T 20438.3—2017。

嵌入式软件的验证和测试要求包括:

——宜进行静态分析并形成文件。

——宜进行动态测试并形成文件。在测试过程中,每个子程序(子程序或函数)宜至少调用一次(入口)。

——动态测试过程中,代码中的所有声明宜至少执行一次。

——如果在诊断功能中使用软件来控制随机硬件故障,则动态测试宜解决诊断的正确实施问题,例如通过故障插入测试。

——动态测试宜包括对目标硬件的最终测试。

7.11 归档

从系统需求提出至完成软件验证计划,整个软件开发过程宜可追溯。

每个软件开发生命周期阶段的文件都要归档,并提供给相关人员。

测试结果和采取的纠正措施宜形成文件。

7.12 配置和变更管理

软件的任何修改或变更都宜进行影响性分析,以确定所有受影响的软件部分,并进行必要的重新设计、重新审查和重新测试活动,以确认其仍符合相关软件需求规范。

宜规定并记录配置管理和变更管理过程文件,且至少宜包括下列项目:

——配置管理要素,至少包含:软件需求、概要和详细的软件设计、软件单元源代码、验证测试的计划、过程和结果;

——每个软件单元或配置要素的唯一识别标识规则;

——从申请到实施的所有变更过程。

对于配置的每个要素,需可识别已发生的所有变更及相关要素的版本。

注 1:目的是能够跟踪每个要素的开发过程(即已经做了哪些修改,为什么修改,何时修改)。

软件配置管理宜允许获得精确且唯一的软件版本标识。配置管理宜关联所有必要的要素(及其版本),以证明主功能的完整性。

在进行最终软件版本评估测试之前,软件配置中的所有要素都宜包含在配置管理过程中。

注 2:此处的目的是确保在软件执行评估程序时,所有要素都处于精确管理的状态。任何后续的变更都可能需要对软件进行修订,以便分析人员可以识别该软件。

宜建立软件及其相关数据的归档程序(存储备份和归档的方法)。

注 3:这些备份和存档可用于在软件的生命周期内维护和修改软件。

7.13 与设备及系统相关的验证

验证宜包括:

——验证在目标硬件上执行时软件指定的功能行为和性能标准(例如时序性能),包括用户用例;

——验证软件措施是否有效降低风险评估中确定的风险;

——验证软件开发过程中为避免系统性软件故障而采取的措施和方法。

第一步,检查是否有嵌入式软件相关的规范和设计文档。宜对规范及文档进行审查,确保其完整

性，检查是否存在错误的解释、遗漏或不一致之处。

注：对于小型程序，通过使用如软件文档（控制流程图、软件单元或块的源代码、I/O 和变量分配列表、交叉引用列表）的方式对控制流、过程等进行审查或走读来对程序进行分析就足够了。

通常，可以将软件视为“黑盒”或“灰盒”，并分别通过黑盒或灰盒测试进行验证。

测试内容包括：

——功能行为和性能的黑盒测试（例如时序性能）；

——基于极值分析的附加扩展测试用例；

——I/O 测试，以确保正常使用输入和输出信号；

——模拟预先分析确定的故障及预期响应的测试用例，以评估基于软件的故障控制措施的充分性；

——从内部和外部客户的角度进行最终用户环境测试，以验证该解决方案是否按预期为最终用户工作。这可以是设备或整个系统验证的一部分。

后续测试宜包括：

——工业网络环境和控制系统测试；

——大量的网络流量测试；

——鲁棒性行为测试；

——测试重复性测试的性能，例如配置写入和电源循环；

——从用户角度测试固件升级性能。

已验证的单个软件功能不需要再次验证。但是，如果为一个特定项目组合了多个这样的功能块，宜验证主功能的最终完整性。同时宜进行影响性分析，以确定哪些软件功能需要重新验证。

宜检查软件相关用户手册，确认已经采取充分的措施和行动来避免软件系统故障。

宜审查软件实施、配置和变更管理措施，以确保其正确执行。

如果嵌入式软件后续进行了修改，则宜在适当的范围内对其进行重新验证。

参 考 文 献

[1] GB/T 2900.99—2016 电工术语 可信性(IEC 60050-192:2015,IDT)

[2] GB/T 8566—2007 信息技术 软件生存周期过程 (ISO/IEC 12207:1995,MOD)

[3] GB/T 16855.1—2018 机械安全 控制系统安全相关部件 第1部分:设计通则(ISO 13849-1:2015,IDT)

[4] GB/T 19898—2005 工业过程测量和控制 应用软件文档集(IEC 61506:1997,IDT)

[5] GB/T 20438(所有部分) 电气/电子/可编程电子安全相关系统的功能安全[IEC 61508(所有部分)]

[6] GB/T 20438.3—2017 电气/电子/可编程电子安全相关系统的功能安全 第3部分:软件要求(IEC 61508-3:2010,IDT)

[7] GB/T 20438.4—2017 电气/电子/可编程电子安全相关系统的功能安全 第4部分:定义和缩略语(IEC 61508-4:2010,IDT)

[8] GB/T 20438.7—2017 电气/电子/可编程电子安全相关系统的功能安全 第7部分:技术和措施概述(IEC 61508-7:2010,IDT)

[9] GB/T 21109.1—2007 过程工业领域安全仪表系统的功能安全 第1部分:框架、定义、系统、硬件和软件要求(IEC 61511-1:2003,IDT)

[10] GB 28526 机械电气安全 安全相关电气、电子和可编程电子控制系统的功能安全(GB 28526—2012, IEC 62061:2005,IDT)

[11] GB/T 34924—2017 低压电气设备安全风险评估和风险降低指南(IEC Guide 116:2010,IDT)

[12] ISO 27005 Information technology—Security techniques—Information security risk management

[13] ISO/IEC TS 17961:2013 Information technology—Programming languages, their environments and system software interfaces—C secure coding rules

[14] ISO/IEC/IEEE 26512:2018 Systems and software engineering—Requirements for acquirers and suppliers of information for users

[15] IEC 60880:2006 Nuclear power plants—Instrumentation and control systems important to safety—Software aspects for computer-based systems performing category A functions

[16] IEC 62443 (all parts) Industrial communication networks—Network and system security

[17] IEC TS 63208:2020 Low-voltage switchgear and controlgear—Security aspects

[18] UL 489:2016 Supplement SE, Molded-case circuit-breakers and molded-case switches with software in programmable components

[19] ANSI/UL 1998:2013 Standard for Software in Programmable Components

[20] MISRA C:2012 Guidelines for the use of the C language in critical systems

[21] CERT/CC C:2016 SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems