ICS 43.020

T40

中华人民共和国国家标准

GB/T XXXXX—XXXX

道路车辆 功能安全

第10部分：指南

Road vehicles — Functional safety — Part 10:Guideline

(ISO26262-10:2012, Road vehicles — Functional safety — Part 10:

Guideline on ISO 26262，MOD)

（征求意见稿）

（完成时间：2016.5.18）

XXXX-XX-XX 发布　　XXXX-XX-XX 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

前　言

GB/T XXXXX《道路车辆 功能安全》包括十个部分：

——第1部分：术语；

——第2部分：功能安全管理；

——第3部分：概念阶段；

——第4部分：产品开发：系统层面；

——第5部分：产品开发：硬件层面；

——第6部分：产品开发：软件层面；

——第7部分：生产和运行；

——第8部分：支持过程；

——第9部分：汽车安全完整性等级导向和安全导向分析；

——第10部分：指南。

本部分为GB/T XXXXX的第10部分。

本部分按照GB/T 1.1-2009给出的规则起草。

本部分修改采用国际标准ISO 26262-10：2012《道路车辆 功能安全 第10部分：指南》（英文版）。

本部分的附录A、B为资料性附录。

本部分由国家标准化管理委员会提出。

本部分由全国汽车标准化技术委员会（SAC/TC114）归口。

本部分起草单位：

本部分主要起草人：

引 言

ISO 26262 是以 IEC61508 为基础，为满足道路车辆上特定电子电气系统的需求而编写。

ISO 26262 适用于道路车辆上特定的由电子、电气和软件组件组成的安全相关系统在安全生命周期内的所有活动。

安全是未来汽车发展的关键问题之一，不仅在驾驶辅助和动力驱动领域，而且在车辆动态控制和主被动安全系统领域，新的功能越来越多的触及到系统安全工程领域。这些功能的开发和集成将强化对安全相关系统开发流程的需求，并且要求提供满足所有合理的系统安全目标的证明。

随着技术日益复杂、软件内容和机电一体化应用不断增加，来自系统性失效和硬件随机失效的风险逐渐增加。ISO 26262 通过提供适当的要求和流程来避免风险。

系统安全是通过一系列安全措施实现的。安全措施通过各种技术（例如，机械、液压、气压、电子、电气、可编程电子等）实现且应用于开发过程中的不同层面。尽管 ISO 26262 针对的是电子电气系统的功能安全，但是它也提供了一个框架，在该框架内可考虑基于其它技术的与安全相关系统。ISO 26262：

a) 提供了一个汽车安全生命周期(管理、开发、生产、运行、维护、报废)，并支持在这些生命周期阶段内对必要活动的剪裁；

b) 提供了一种汽车特定的基于风险的分析方法以确定汽车安全完整性等级（ASIL）；

c) 应用汽车安全完整性等级规定 ISO 26262 中适用的要求，以避免不合理的残余风险；

d) 提供了对于确认和认可措施的要求，以确保达到一个充分、可接受的安全等级；

e) 提供了与供应商相关的要求。

功能安全受开发过程(例如包括需求规范、设计、实现、集成、验证、确认和配置)、生产过程、维护过程和管理过程的影响。

安全问题与常规的以功能为导向和以质量为导向的开发活动及工作成果相互关联。ISO 26262 涉及与安全相关的开发活动和工作成果。

图1为ISO 26262的整体架构。ISO 26262基于V模型为产品开发不同阶段提供过程参考：

——阴影”V”表示标准中第 3、4、5、6 和 7 部分之间的关系；

——以“m-n”方式表示的具体条款中，“m”代表特定部分的编号，“n”代表该部分章条的编号。

示例：“2-6”代表 ISO 26262-2 的第六章

图1 ISO 26262概览

道路车辆 功能安全
第10部分：指南

1 范围

GB/T XXXXX适用于安装在最大总质量不超过3.5吨的量产乘用车上的包含一个或多个电子电气系统的与安全相关的系统。

GB/T XXXXX不适用于安装在特殊用途车辆上的特定的电子电气系统，例如为残疾驾驶者设计的车辆。

已经完成生产发布的系统及其组件或在本标准发布日期前开发的系统及其组件不适用于本标准。对于在本标准发布前完成生产发布的系统及其组件进行进一步的开发或变更时，仅修改的部分需要按照本标准开发。

本标准针对由电子电气安全相关系统的故障行为而引起的可能的危害，包括这些系统相互作用而引起的可能的危害。本标准不针对与触电、火灾、烟雾、热、辐射、毒性、易燃性、反应性、腐蚀性、能量释放等相关的危害和类似的危害，除非危害是直接由电子电气安全相关系统的故障行为而引起的。

本标准不针对电子电气系统的标称性能，即使这些系统（例如主动和被动安全、制动系统、自适应巡航系统）有专用的功能性能标准。

本部分提供了GB/T XXXXX的概览，也给出了额外的解释，目的是增强对GB/T XXXXX其它部分的理解。本部分只具有资料性特性，描述了GB/T XXXXX的一般概念以便于理解。该解释将一般概念扩展到特定的内容。

如果本部分与GB/T XXXXX其它部分存在不一致时，以GB/T XXXXX其它部分中定义的要求、建议和信息为准。

2 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是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 XXXXX-1:201X，道路车辆 功能安全 第1部分：术语；

GB/T XXXXX-2:201X，道路车辆功能安全 第2部分：功能安全管理；

GB/T XXXXX-3:201X，道路车辆功能安全 第3部分：概念阶段；

GB/T XXXXX-4:201X，道路车辆功能安全 第4部分：产品开发：系统层面；

GB/T XXXXX-5:201X，道路车辆 功能安全 第5部分：产品开发：硬件层面；

GB/T XXXXX-6:201X，道路车辆功能安全 第6部分：产品开发：软件层面；

GB/T XXXXX-7:201X，道路车辆功能安全 第7部分：生产和运行；

GB/T XXXXX-8:201X，道路车辆功能安全 第8部分：支持过程；

GB/T XXXXX-9:201X，道路车辆功能安全 第9部分：以汽车安全完整性等级为导向和以安全为导向的分析；

3 术语、定义和缩略语

GB/T XXXXX-1给出的术语、定义和缩略语适用于本部分。

4 GB/T XXXXX 中的关键概念

4.1 针对汽车系统的功能安全（与 GB/T 20438 的关系）

GB/T 20438《电气/电子/可编程电子安全相关系统的功能安全》是关于电气、电子、可编程电子安全相关系统的功能安全通用标准和基础安全标准。这意味着，各工业领域将基于GB/T 20438的要求建立自身的功能安全标准。

在汽车行业，如果直接应用GB/T 20438标准会存在许多问题。其中的一些问题和GB/T XXXXX中相应的差异描述如下。

GB/T 20438基于“受控设备”模型，例如具有如下关联控制系统的工业工厂：

a) 危害分析识别出与受控设备（包括设备控制系统)关联的危害，并对此应用风险降低措施。这可通过电子/电气/可编程电子系统、其它技术的安全相关系统（例如：安全阀)、或外部措施（例如：工厂的物理围堵）来实现。GB/T XXXXX基于严重度、暴露概率和可控性，为危害分级提供了规范化的汽车领域的方案。

b) 分配给电子/电气/可编程电子系统的风险降低，是通过指定的安全功能实现的。这些安全功能要么是独立的保护系统的一部分、要么集成到工厂控制中。但在汽车系统中，未必能做出这种区分。车辆的安全依赖于控制系统自身的表现。

GB/T XXXXX使用了安全目标和安全概念的如下思想：

——通过危害分析和风险评估识别出需要防止、减轻或控制的危害和危害事件；

——为每个危害事件制定安全目标；

——将汽车安全完整性等级（ASIL）与每个安全目标关联；

——功能安全概念是实现安全目标的功能性的表述；

——技术安全概念是功能如何通过硬件和软件在系统层面实现的表述；

——软件安全要求和硬件安全要求表述了特定的安全要求，这些要求将作为软硬件设计的一部分而被实施。

示例：

——安全气囊系统：其中一个危害是气囊非预期的起爆。

——相关的安全目标是气囊不能起爆，除非发生需要气囊起爆的碰撞。

——功能安全概念可定义冗余的功能来探测车辆是否发生碰撞。

——技术安全概念可定义两个具有不同轴向的独立加速度传感器和两个独立点火回路的实施方案，如果两路均闭合则起爆点火管。

GB/T 20438针对的是单一的或小批量的系统。系统经过制造和测试后安装在工厂，然后执行安全确认。对于大批量销售的系统，如道路车辆，安全确认在量产之前执行。因此GB/T XXXXX中生命周期活动的次序有所不同。对此，GB/T XXXXX-7提出了对生产的要求。而在GB/T 20438中并未覆盖此部分内容。

GB/T 20438未对管理跨多个组织和供应链的开发提出特定要求，而GB/T XXXXX明确的提出了这个问题，包括开发接口协议(DIA)（参见GB/T XXXXX-8，第5章（分布式开发的接口)），这是因为汽车系统是由客户（如整车厂、客户的供应商或客户自己）的一个或多个供应商生产的。

GB/T 20438不包含危害分级的规范化要求。GB/T XXXXX则包含了汽车领域危害分级的方案。此方案认为不是汽车系统的每个危害都会导致事故。其结果取决于涉险人员是否实际暴露在危

害发生的场景下，且它们能否采取措施以控制危害的结果。图2给出了应用了此概念的失效示例，即：影响运动中车辆可控性的失效。

注：此概念仅为了阐述失效的发生和事故之间没有必然的直接关联。尽管此过程中评估的参数与图中状态过渡的可能性相关，但它不是危害分析和风险评估过程的展示。

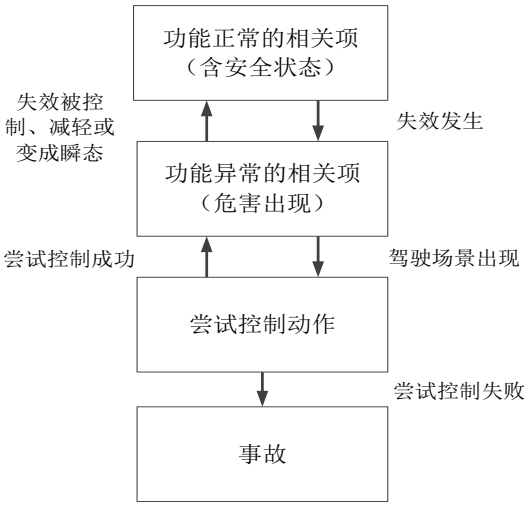


图 2 汽车风险的状态机模型

为符合汽车工业的当前技术水平，对硬件开发（GB/T XXXXX-5）和软件开发（GB/T XXXXX-6）要求进行了调整。特别地，GB/T XXXXX-6包含涉及基于模型开发的要求；GB/T 20438规定了特定方法的应用，对于使用任何替代方法，必须提供详细的理由。而对于GB/T XXXXX中列出方法，给出了特定的目标。为实现这些目标，可应用给出的方法，或应用替代方法，但需提供理由证明替代方法也能实现目标。

为GB/T XXXXX中的安全要求分配汽车安全完整性等级(ASIL)而不是安全完整性等级(SIL)，其主要原因是IEC 61508中的SIL表述为概率术语（参见IEC 61508-1：2010，表3）。IEC 61508阐述:“在评估目标的失效措施是否达成时，可以接受仅考虑硬件安全完整性的量化技术及应用可靠性预测技术。为满足（有关系统性安全完整性的）目标失效措施的必要预防措施，须使用定性技术和判断”。ASIL不是基于这类有关危害发生的概率的要求；但也存在关于符合ASIL要求的概率目标。

4.2 相关项、系统、要素、组件、硬件元器件和软件单元

GB/T XXXXX-1 中定义了相关项、系统、要素、组件、硬件元器件和软件单元。图 3 展示了系统、要素、组件、硬件元器件和软件单元的关系。图 4 举例展示了相关项的分解（构成关系）。可分解的要素可被标注为系统、子系统或组件。满足系统标准的可分解要素可被标注为系统或子系统。当着重强调要素是一个更大系统的一部分时，使用术语“子系统”。组件是非系统层面的、逻辑上和技术上独立的要素。通常，术语“组件”用于仅由元器件和单元组成的要素，但也能用于由更低层面的特定技术领域（例如：电气/电子技术（参见图 4））要素组成的要素。

示例：对于微控制器或特定用途集成电路（ASIC），可使用如下分割：整个微控制器是一个组件，处理单元（例如：中央处理单元（CPU））是一个元器件，处理单元中的寄存器（例如：中央处理单元寄存器组）是一个子元器件。在进行微控制器分析时，需要进行更细节层面的分割，为帮助达到此目的，可能要将元器件分割成子元器件，再将子元器件进一步分割成基础子元器件。

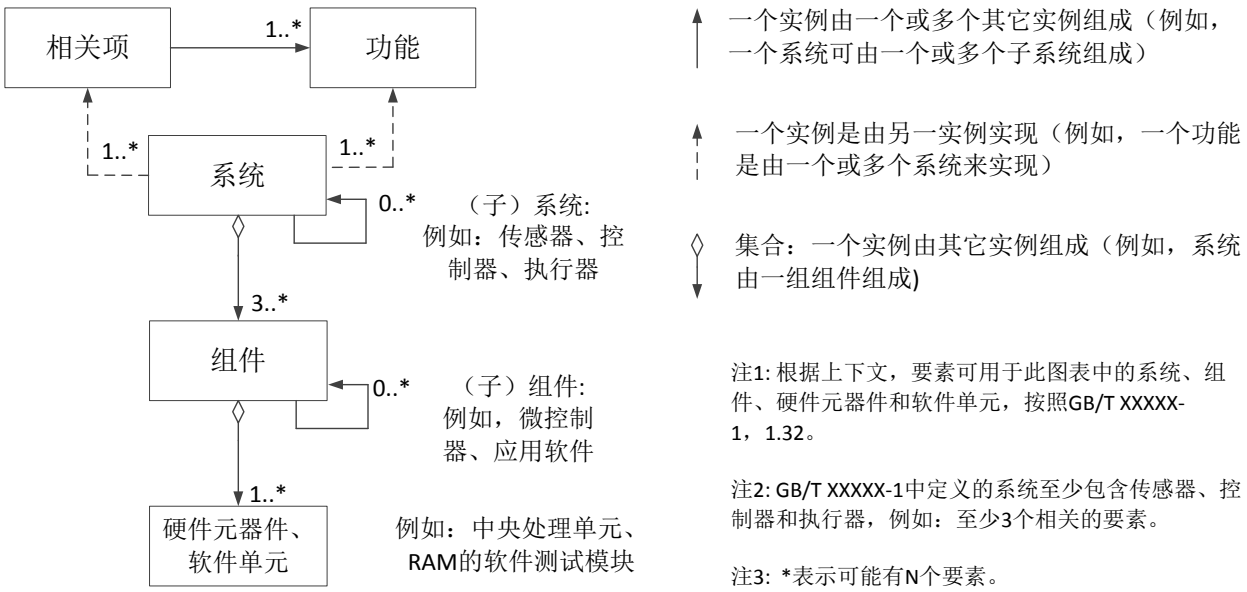


图 3 相关项、系统、组件、硬件元器件和软件单元间的关系

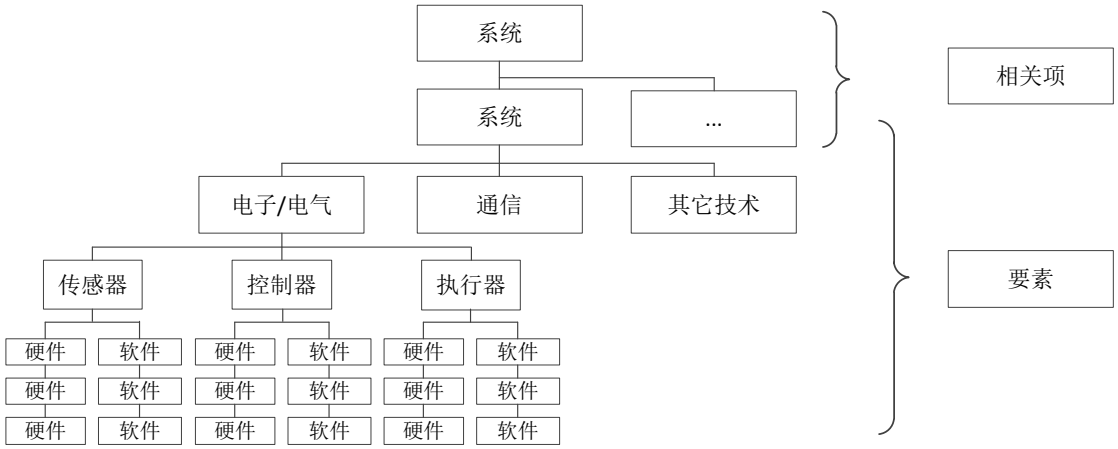


图 4 相关项分解示例

4.3 故障、错误和失效之间的关系

在 GB/T XXXXX-1 中定义了术语：故障、错误和失效。图 5 从三个不同类型的原因（系统性软件问题、随机硬件问题和系统性硬件问题）描述了故障到错误并从错误到失效的发展过程。系统故障（参见 GB/T XXXXX-1）起因于设计和规范的问题；软件故障和部分硬件故障是系统性的。随机硬件故障（参见 GB/T XXXXX-1）起因于物理过程，比如疲劳、物理退化或环境应力。在组件层面，每个不同类型的故障会导致不同的失效。然而，组件层面的失效是相关项层面的故障。注意，在此示例中，整车层面不同原因导致的故障可引起相同的失效。如果额外的环境因素使失效叠加了事故场景，相关项层面的部分失效将会是危害（参见 GB/T XXXXX-1）。

示例：当车辆开始穿越十字交叉路口时发生非期望的行为，可能发生碰撞，例如：对危害事件“当车辆开始穿越十字交叉路口时发生猛烈窜动”的风险进行严重度、暴露概率和可控性的评估（“猛烈窜动”是指突然的剧烈移动）。

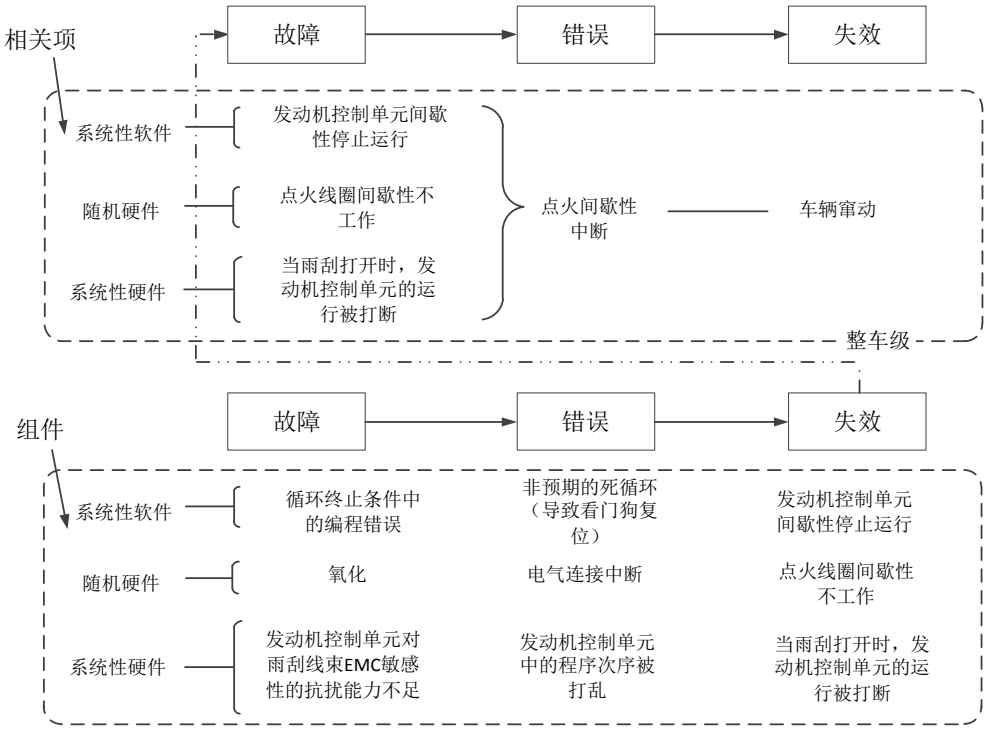


图 5 故障导致失效的示例

5 关于安全管理的精选话题

5.1 工作成果

本子章节描述术语“工作成果”。

工作成果是满足GB/T XXXXX相应要求的结果（参见GB/T XXXXX-1）。因此，文档化的工作成果可提供符合这些安全要求的证据。

示例：需求规范是可通过需求数据库或文本文件来记录的一种工作成果。可执行模型是能通过可执行建模语言文件表示的一种工作成果，例如，通过使用软件工具达到仿真的目的。

工作成果的文档（参见GB/T XXXXX-8第10章）作为已执行的安全活动、安全要求或相关信息的记录，不局限于任何形式或媒介。

示例：工作成果的文档可通过电子或纸质文件表示，通过单一或系列文档表示。它可以与其它工作成果的文档合并，或与非功能安全直属文档合并。

为避免信息重复，可在文档内或文档间使用交叉引用。

5.2 认可措施

5.2.1 总则

GB/T XXXXX 中定义的工作成果，或作为认可措施的一部分，或作为验证活动的一部分，在后续活动中得到评估。本子章节描述了验证和认可措施间的差异。

一方面，开展验证活动是为了确定安全要求的完整性，及安全要求的定义和实施的正确性。

工作成果的验证可包括：

——验证评审，对比更高层面的安全要求，验证导出后的安全要求在定义和实施的完整性及正确性；或

——测试案例的执行或测试结果的检查，通过检测相关项或其要素，来提供满足已定义的安全要求的证据。

GB/T XXXXX-3、GB/T XXXXX-4、GB/T XXXXX-5 和 GB/T XXXXX-6 中对验证活动进行了定义。此外, GB/T XXXXX-8 第 9 章(验证)定义了有关验证活动的一般要求，GB/T XXXXX-8 第 6 章(安全要求的定义和管理)定义了对安全要求进行验证的进一步细节。

另一方面，执行认可措施是为了评估相关项功能安全的实现，包括对以下内容的认可：

——依照 GB/T XXXXX 的要求，对相关项开发过程中开展的安全活动和实施的安全过程，进行恰当的定义、剪裁和执行；及

——符合 GB/T XXXXX 相应要求的恰当工作成果。

GB/T XXXXX-2 第 6 章（概念阶段和产品开发过程中的安全管理）定义了认可措施。

示例：如果在系统设计阶段应用 ASIL 分解，则：

——根据技术安全概念（参见 GB/T XXXXX-4, 7.4.8）对生成的系统设计进行验证；及

——依照 GB/T XXXXX-9 第 5 章（关于 ASIL 剪裁的要求分解）对正确实施 ASIL 分解的确认，可作为功能安全评估的一部分，包括：对已经开展的相关失效分析的确认和对声明执行相应冗余安全要求的要素间具备充分独立性的论证。

5.2.2 功能安全评估

如果相关项安全目标的最高 ASIL 等级是 ASIL C 或 D，则开展功能安全评估以评价相关项是否实现功能安全。在 GB/T XXXXX-2 中，分别描述了功能安全评估的某些方面，即功能安全审核和认可评审。

功能安全评估包括：

a) 对已实施的、可在相关项开发过程中评估的安全措施的恰当性和有效性的评审；

b) 对安全计划所要求的工作成果的评估。着重对选定的工作成果进行评审。这些活动统归为认可评审，目的是确认这些工作成果符合 GB/T XXXXX 的相应要求；及

c) 一次或多次功能安全审核，以评估功能安全所要求的流程的实施情况。

功能安全评估可被重复或更新。

示例 1：因变更管理流程识别出相关项或其要素的变更对相关项的功能安全存在影响[参见 GB/T XXXXX-8，第 8 章（变更管理）]，而对功能安全评估进行更新。

示例 2：因功能安全评估报告包含了对相关项功能安全进行“有条件接受”或“拒绝”的建议，由此需要重复进行功能安全评估。在此情况下，重复评估包含对上一次功能安全评估所做建议的跟进，如果适用，还包括对已实施的纠错行动的评价。

如果相关项安全目标的最高 ASIL 等级是 ASIL B，功能安全评估可被省略或以不严格的方式执行。然而，即使不执行功能安全评估，仍需执行其它认可措施，即：对危害分析和风险评估、安全计划、相关项集成与测试计划、确认计划、适用的安全分析、在用证明（如果适用）和安全档案的完整性的认可评审（参见 GB/T XXXXX-2，表 1）。

如果相关项安全目标的最高 ASIL 等级是 ASIL A，GB/T XXXXX 不要求、不建议、但也不反对执行功能安全评估。然而，仍需执行对危害分析与风险评估、及适用的安全分析的认可评

审。

在分布式开发的情况下，功能评估的范围包括由整车厂和相关项供应链中的供应商生成的工作成果、实施的流程及安全措施[参见 GB/T XXXXX-2 和 GB/T XXXXX-8，第 5 章（分布式开发的接口）]。

功能安全评估的目的是评价相关项功能安全的实现，这只能在相关项层面进行。因此，在（开发相关项要素的）供应商处所作的功能安全评估具有局限性，从本质上它是后续（客户层面）功能安全评估活动的输入。作为相关项开发中的最终客户，整车厂指派人员开展完整范围内的功能安全评估，以判断相关项功能安全的实现。此判断包括对相关项的功能安全提供“接受”、“有条件接受”和“拒绝”的建议。

注：对于由一级供应商负责包括整车集成的相关项开发的情况，该供应商承担整车厂的上述角色。

实际方法上，分布式开发中的功能安全评估可就此分解为：

——在供应商处的有限范围的功能安全评估，涉及供应链中的供应商。适用的 ASIL 等级是供应商开发的相关项各要素所继承的（相关项各安全目标的）最高 ASIL 等级(参见 GB/T XXXXX-8，5.4.5)；及

——最终的功能安全评估，包括对集成的相关项实现功能安全的判断，例如由整车厂开展的评估。适用的 ASIL 等级是相关项全部安全目标中最高的 ASIL 等级(参见 GB/T XXXXX-2)。

示例：整车厂开发一个具有 ASIL D 安全目标（SG1）和 ASIL A 安全目标（SG2）的相关项，并将对其开展功能安全评估。存在如下可能情况，某二级供应商或三级供应商只开发了相关项的 ASIL A 要素，即仅继承 SG2 的 ASIL 等级的要素[然而，如果适用，参考 GB/T XXXXX-9，第 6 章（要素共存的准则）]。对于在该供应商处开展有关此相关项开发的功能安全评估，GB/T XXXXX 未给出（支持或反对的）要求或建议。

与客户和供应商接口有关的功能安全评估的范围、评估的流程（例如：需由供应商提供的工作成果，需由客户评审的工作成果）和评估的执行，在相应开发接口协议中进行了定义[参见 GB/T XXXXX－8，第 5 章(分布式开发的接口)]。

示例：整车厂（客户）与一级供应商之间的开发接口协议（DIA）。一级供应商与二级供应商之间的开发接口协议（DIA）。

在分布式开发的情况下，功能安全评估的可能开展方式是整车厂和供应链中的供应商各自处理所负责的评估活动（参见上述 a)、b)、c)项）的如下方面：

——供应商对所开发要素中实施的安全措施进行评审，包括措施对满足相应安全目标或安全要求（由客户提供或由供应商开发）的适当性和有效性，并评估安全措施的实施过程及适用的工作成果。供应商也评估所开发要素对相关项功能安全的潜在影响，例如：对实施的可带来新危害的安全措施的识别；及

——整车厂对集成后的相关项的功能安全进行评估。评估的一部分可基于由一个及以上供应商提供的工作成果或信息，包括在供应商处开展的功能安全评估的报告。

注：客户可对供应商实施的安全措施及完成的工作成果进行评估。客户也可对供应商处的实施过程进行评估（参见 GB/T XXXXX-8，5.4.4.8）。

5.3 安全档案的理解

5.3.1 安全档案的解释

安全档案的目的是提供一个由证据支持的、清晰的、全面的和正当的论据，以证明当运行在预期的环境中时，相关项不存在不合理的风险。

此处提供的指南聚焦于 GB/T XXXXX 的范畴。

安全档案有三个主要素：

——要求；

——论证；及

——证据，即 GB/T XXXXX 的工作成果。

图 6 描述了 GB/T XXXXX 上下文中这三个要素间的关系。

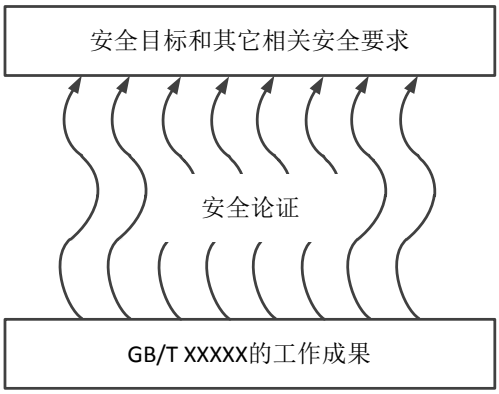


图6 安全档案（参见[2]）的关键要素

安全论证表达了证据和目的之间的关系。安全论证的重要性经常被忽视。可能展示了许多页的支持证据却没有清晰的解释此证据如何联系到安全目的。论证和证据都是安全档案的决定性要素，并且是齐头并进的。没有支持证据的论证是没有事实根据的，因此是不足以令人相信的。没有论证的证据是不清楚的，以致缺乏安全目的如何得到满足的清晰解释。通过开发和展示安全档案报告，表达安全档案。安全档案报告的任务是汇总安全论证，并对记录安全支持证据的报告进行引用（例如：测试报告）。

在其它行业中使用的安全论证，经常是通过叙述性文本的安全案例报告表达的。叙述性文本可描述安全目的是如何被解释、分配和分解的，最终引用证据证明符合较低层面的安全声明。或者，使用图形化论证符号（如："声明-论证-证据"和目标结构表示法[2]）正变得越来越流行，形象化的和明确的展示安全论证的独立要素（要求、声明、证据和背景），同时展示这些要素间存在的关系（即，特定声明如何支持独立要求，证据和为论证定义的假设背景如何支持声明）。

通过直接诉诸实施的相关项的特征（例如：定时看门狗的行为）来证明安全的安全论证通常被称为产品论证。通过诉诸开发和评估过程的特征（例如：采用的设计符号）来证明安全的安全论证通常被称为过程论证。

可使用两种类型的论证，以达到对相关项安全的完整论证，这里，过程论证可被看成是为产品论证中用到的证据提供可信度。

5.3.2 安全档案开发生命周期

安全档案的开发可被视为与安全生命周期内其余开发阶段集成的增量活动。

注：安全计划可包含安全档案增量步骤的计划和安全档案初始版本的计划。

这种方法允许在产品开发的给定节点生成安全档案的中间版本。例如：安全档案的初始版本可在技术安全要求验证后生成；安全档案的中间版本可在系统设计验证后生成；及最终版本可在功能安全评估前生成。

安全档案需符合 GB/T XXXXX-2 6.4.7（认可措施：类型、独立性和权限）中给定的认可评审。

如果相关项被修改，需评估对安全档案的影响，如果必要，需根据修改对安全档案进行更新。

6 概念阶段和系统开发

6.1 总则

本节通过使用简单的示例，为概念阶段提供了危害分析和风险分级的原理概览。

6.2 危害分析和风险评估示例

6.2.1 总则

考虑以控制车载嵌入式能量存储装置的相关项为示例。关于此示例的用途，只有在车辆行驶速度大于等于 15km/h 时才试图释放储存的能量。如果车速低于 15km/h，储存能量的释放会导致过热进而引发装置爆炸。

6.2.2 分析 1

a) 危害识别

——导致装置中能量非预期释放，从而产生爆炸的失效

b) 危害事件

关于此示例的用途，进行危害分析和风险评估所考虑的驾驶场景为：

——交通堵塞中行驶速度低于 15km/h

相关项发生失效导致非预期的能量释放，储能装置爆炸，对车辆乘员造成严重伤害。

c) 对识别出的危害事件分级

爆炸对车辆中的乘员导致危及生命的伤害，存活不确定，因此，危害严重度预估为 S3。

车辆处在交通堵塞中，且车速低于 15km/h。基于对车辆目标市场的交通统计，此场景的暴露概率预估为 E3（1%-10%的驾驶时间内发生）。

车辆驾驶员或乘客控制相关项失效和装置爆炸的能力被认为是不可能的：可控性预估为 C3（难以控制或不可控）。

应用 GB/T XXXXX-3，表 4：ASIL 等级确定为 ASIL C。

6.2.3 分析 2

a) 危害识别

——未导致能量释放的失效

b) 危害事件

——任何驾驶场景

相关项发生失效但不导致能量从储能装置中释放，所以不产生伤害。

c) 对识别出的危害事件分级

由于相关项的失效不会导致伤害，严重度评为 S0，无需确定可控性，也无需定义安全目

标。

6.3 关于可控性分级

如 GB/T XXXXX-3，第 7 章（危害分析和风险评估）中的解释，可控性代表对驾驶员或其它交通参与者能够避免特定伤害的能力的预估。

在最简单的情况下，只考虑给定危害事件的一个后果，可控性代表了对避免此后果的可能性的预估。然而，也可能存在其它情况。例如，可能有一个严重后果（例如：严重度 S2），却很容易避免（例如：可控性 C1）；又或者后果的严重度较低（例如：S1），却难以避免（例如：C3）。假设暴露概率等级为 E4，下面几组数值是可能的结果，其说明了最高的严重度导致最高的 ASIL 等级不是必然成立的。

——E4, S2, C1 => ASIL A

——E4, S1, C3 => ASIL B

在此示例中，ASIL B 是危害事件的合理分级。

6.4 外部措施

6.4.1 总则

外部措施是独立于且不同于相关项的措施，用于降低或减轻相关项失效造成的风险。

6.4.2 基于车辆的外部措施示例 1

车辆 A 装备手动变速箱，当熄火后，变速箱能处于任何档位，包括空档。车辆 B 装备自动变速箱，熄火状态下，维持一个档位啮合且离合器常闭状态。两辆车都有附加相关项，电子驻车制动（EPB）。

对两辆车辆都分析的场景含：

——车辆处于驻车状态（熄火，驾驶员不在）。

——在人口稠密的城市区域、有坡度的路边驻车。

——发生涉及电子驻车功能突然丧失的失效。

在此场景中，对于车辆 A，熄火时处于空档（对应于合理可预见的误用的场景），如果无人看管，车辆将可能发生移动。这会导致评估的可控性为 C3、严重度为 S2 或更高（这取决于车辆附近是否有容易受伤的人员）且暴露概率等级大于 E0。按照实际分配的暴露概率等级，得出的 ASIL 等级在 QM 和 C 之间。

车辆 B 一直结合在档位上而不会发生移动，所以不会导致危害。此设计中包含的基于车辆的外部措施有助于消除该场景下的风险，但前提是自动变速器和电子驻车系统能被证明是充分独立的。

6.4.3 基于车辆的外部措施示例 2

车辆 A 装备动态稳定控制系统和启停功能。车辆 B 只装备了启停功能。

对两辆车辆都分析的场景包含：

——车辆以中高速行驶（50km/h<v<90km/h）。

——路面平坦、干燥，且在郊区。

——车辆正在接近一个中等曲率的道路。

——车速和道路曲率会产生中高侧向加速度。

——启停功能中的失效引起发动机非预期熄火，导致在此场景中突然失去驱动力。

作为突然丢失牵引力的后果，车辆会产生横摆力矩，这要求驾驶员调整方向盘以重新控制车辆。在车辆 B 上执行这个动作可被证明具有更低的可控性，导致 C 或 D 的 ASIL 分级。相比之下，车辆 A 的动态稳定性控制功能会限制侧向不稳定的影响。结果是，对于车辆 A，降低了可控性的级别。因此，由动态稳定性控制实现的基于车辆的外部措施有助于此场景下的风险降低。然而，仅当能证明所考虑的启停功能失效不会影响动态稳定性控制功能，并且对这两个功能来说不是关联失效，才适用这种情况。

6.5 合并安全目标的示例

6.5.1 介绍

安全目标是相关项的顶层安全要求。它们导出避免危害事件产生不合理风险所需的功能安全要求。按照 GB/T XXXXX3，7.4.8，概念阶段中会确定安全目标。当安全目标相似或涉及不同场景下的相同危害时，它们可以被合并成以原始安全目标的最高 ASIL 等级为最终 ASIL 等级的一个安全目标。这使管理的安全目标变少，却覆盖了所有识别出的危害，因此可简化后续的开发活动。

6.5.2 总则

下面示例中展示的相关项、安全目标和 ASIL 分级仅为了说明安全目标合并的过程。该示例不能反映将 GB/T XXXXX 应用到相似真实项目的情况。特别是，它没有完整表述失效模式识别、场景分析和整车层面的影响评估。

为简单起见，示例只限于两个安全目标的合并，但相同的方法可以扩展到更多初始安全目标的合并。

6.5.3 功能定义

考虑车辆装备了电子驻车制动（EPB）系统。当被驾驶员的特定请求激活时，EPB 系统在车辆后轮施加制动力矩以防止驻车时车辆非预期的移动。

6.5.4 应用于不同场景下相同危害的安全目标

6.5.4.1 危害分析和风险评估

为简化示例，考虑驻车功能的如下失效模式：

——非预期的驻车制动激活。

注：在此上下文中，术语“非预期的激活”是指在没有驾驶员请求的情况下的功能动作。

根据故障发生时的特定场景，此失效模式会导致不同的车辆影响，如表 1 所示。

表 1 由不同场景下相同危害得出的安全目标

失效模式	危害	特定场景	危害事件	可能结果	ASIL	安全目标	安全状态
非预期的驻车制动激活	非预期的减速	高速行驶或转弯或低附路面	高速行驶或转弯或低摩擦路面时非预期的减速	失去车辆稳定性	较高 ASIL 等级	当车辆移动时，避免在没有驾驶员请求的情况下激活驻车功能	禁 EPB

非预期的驻车制动激活	非预期的减速	中低速行驶且高附路面	中低速行驶且高摩擦路面时非预期的减速	与后车追尾	较低 ASIL 等级	当车辆移动时，避免在没有驾驶员请求的情况下激活驻车功能	禁止EPB

6.5.4.2 安全目标的阐述

如上所示，相同的安全目标和安全状态适用于两种场景。因此，可定义如下安全目标：

——安全目标：当车辆移动时，避免在没有驾驶员请求的情况下激活驻车功能。

——安全状态：禁止 EPB 功能。

——ASIL: 表 1 中确定的较高 ASIL 等级分配给此安全目标。

7 安全过程的要求结构-安全要求的运行和顺序

图 7 和图 8 展示了符合 GB/T XXXXX 的安全要求开发流和顺序，并在下面略述。特定章节以如下方式表示：“m-n”,m 代表部分的数值，n 代表章的数值或子章的数值。

开展危害分析和风险评估以识别风险并为这些风险定义安全目标。[参见 GB/T XXXXX-3，第 7 章（危害分析和风险评估）]

导出的功能安全概念定义了功能安全要求，以满足功能安全目标。这些要求定义了被相关项所使用的安全机制和安全措施。此外，对支持这些要求的系统架构要素进行了识别。[参见 GB/T XXXXX-3，第 8 章（功能安全概念）]

导出的技术安全概念定义了技术安全要求及其对系统要素的分配，以被系统设计所实现。这些技术安全要求将指明硬件要素和软件要素的分割。[参见 GB/T XXXXX-4，第 6 章（定义技术安全要求）]

按照技术安全要求进行系统设计开发。技术安全要求的实施可在系统设计规范中进行定义。[参见 GB/T XXXXX-4，第 7 章（系统设计）]

最后,将提供硬件和软件安全要求以符合技术安全要求和系统设计。[参见 GB/T XXXXX-5，第 6 章（定义硬件安全要求）和参见 GB/T XXXXX-6，第 6 章（定义软件安全要求）]

图 7 说明了 GB/T XXXXX 硬件要求与设计阶段之间的关系

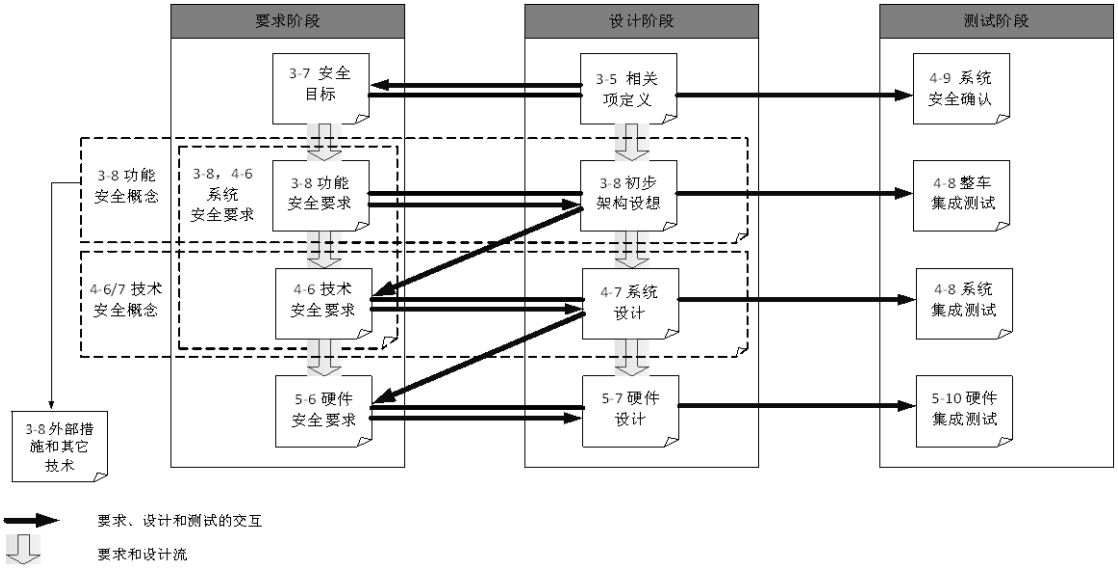


图 7 — 从概念到硬件的安全要求、设计和测试流

图 8 说明了 GB/T XXXXX 软件要求、设计和测试子阶段之间的关系。

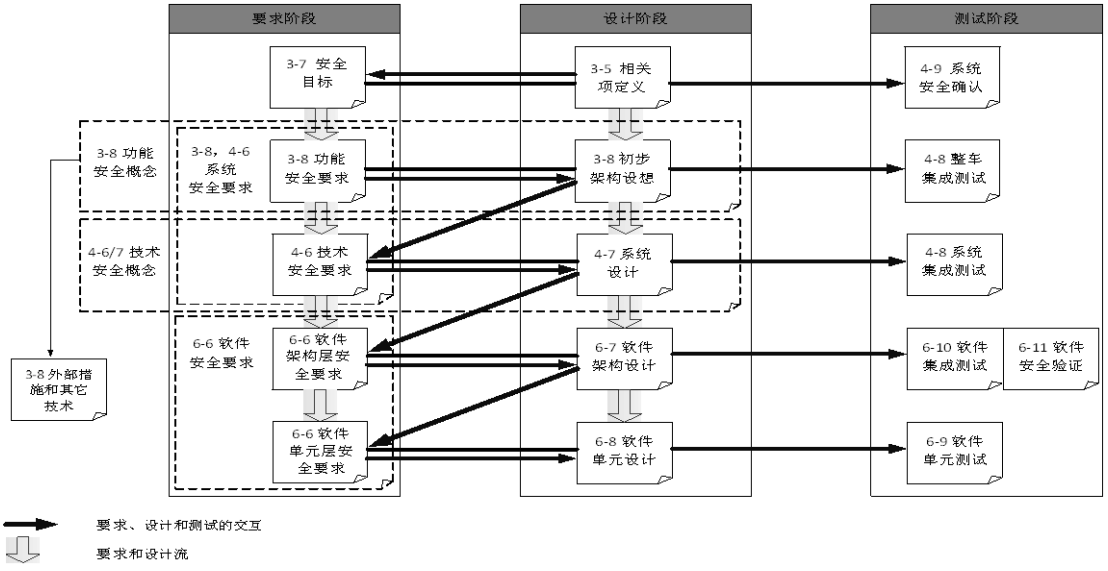


图 8 — 从概念到软件的安全要求、设计和测试流

——系统设计：

从相关项定义（3-5）到初步架构设想、再到系统设计（4-7），不断细化系统设计。

——测试层面间的相关性：

每个层面上的测试规范和测试案例主要取决于相应的要求和设计。它们不取决于其它

测试层面的测试规范、测试案例和测试结果。测试规范通常依赖于测试环境。

——测试层面与要求层面的相关性、测试层面与设计层面的相关性：

测试规范和测试案例由相同层面的要求得出，并由相同层面的设计信息所支持。

示例：对于性能测试，设计信息是必要的。

——软件安全要求验证：

软件安全要求验证阶段(6-11)需要对软硬件进行集成。

——外部措施和其它技术：

在整车层面确认外部措施和其它技术。

8 关于硬件开发

8.1 硬件随机故障的分类

8.1.1 总则

一般来说，所考虑的故障组合限定于两个独立硬件故障的组合，除非基于功能安全概念或技术安全概念的分析显示，n（n>2）点故障是相关的。因此，在大多数情况下，对于给定的安全目标和给定的硬件要素，故障可被归为以下某种类别：

a) 单点故障；

b) 残余故障；

c) 可探测的双点故障；

d) 可感知的双点故障；

e) 潜伏的双点故障；或

f) 安全故障。

下文将给出对于不同故障类别的解释及示例。

8.1.2 单点故障

该故障：

——可直接导致违背安全目标；及

——是硬件要素的故障，对于该硬件要素，没有任何安全机制预防其某些违背安全目标的故障。

示例：一个未被监控的电阻，该电阻至少有一种失效模式（例如：开路）有违背安全目标的潜在可能。

注：如果一个硬件元器件有至少一个安全机制（例如：微控制器的看门狗），则该元器件的故障不被归类为单点故障。对于安全机制未预防其违背安全目标的故障，被归类为残余故障。

8.1.3 残余故障

该故障：

——可直接导致违背安全目标；及

——是硬件要素的故障，对于该硬件要素，有至少一个安全机制预防其某些违背安全目标的故障。

示例：如果仅用棋盘 RAM 测试的安全机制来检查随机存储器(RAM)模块，那么不能探测出某些种类的桥接故障。因这些故障导致的对安全目标的违背不能被安全机制所预防。这些故障即为残余故障的示例。

注：此情况中，安全机制的诊断覆盖率小于 100%。

8.1.4 可探测的双点故障

该故障：

——促使安全目标的违背；

——仅与另一个（双点故障有关的）独立硬件故障联合，才能导致安全目标的违背；及

——被防止其潜伏的安全机制所探测。

示例 1：被奇偶校验保护的闪存：按照技术安全概念对单位故障进行探测并触发响应，如：关闭系统并通过警示灯通知驾驶员。

示例 2：被错误探测和纠错码(EDC)保护的闪存：按照技术安全概念通过测试对这些 EDC 逻辑中的故障进行探测并触发响应，如：通过警示灯通知驾驶员。

对于安全机制使相关项恢复到无故障状态的瞬态故障情况，即使驾驶员从未被通知故障的存在，此故障可被考虑为可探测的双点故障。

示例：在数据提供给 CPU 前，瞬态的位翻转被错误探测和纠错码(EDC)纠正，并通过写回正确值得到后续纠正。可使用（数据）记录来区分间发故障和真正的瞬态故障。

8.1.5 可感知的双点故障

该故障：

——促使安全目标的违背，但仅与另一个（双点故障有关的）独立硬件故障联合，才会导致安全目标的违背；及

——在规定的时间内被驾驶员所感知（有或无安全机制探测）。

示例：如果故障后果显著和清楚的影响功能，双点故障可被驾驶员感知。

注：如果双点故障同时被驾驶员感知并被安全机制探测，该故障可被归类为可探测的双点故障或可感知的双点故障。但它不能同时被归类为这两种类型，因为一个故障如果既是可探测的双点故障、又是可感知的双点故障，那么潜伏故障度量会错误的计算该故障两次。

8.1.6 潜伏的双点故障

该故障：

——促使安全目标的违背，但仅与另一个独立硬件故障联合，才会导致安全目标的违背；及

——不被安全机制所探测也不被驾驶员感知。直到第二个独立故障发生前，系统始终可以运行且驾驶员也不知道发生了故障。

示例 1：对于被 EDC 保护的闪存：在读取时，EDC 纠正了单位的永久性故障值，但这不是在闪存中纠正也无信号指示。在此情况中，故障不能导致安全目标的违背（因故障位已得到了纠正），且它不是可探测的（因对单位故障无信号指示），也不是可感知的（因对应用的功能性无影响）。如果在 EDC 逻辑中发生了额外的故障，它可导致失去对单位故障的控制，从而导致潜在的安全目标的违背。

示例 2：对于被 EDC 保护的闪存：导致 EDC 不可用的 EDC 逻辑故障未被测试探测出来。

8.1.7 安全故障

安全故障可以是以下两类故障中的一种：

a) n > 2 的全部 n 点故障（除非安全概念显示它们与安全目标的违背有关联），或

b) 与安全目标违背无关的故障。

示例 1：对于被 EDC 和循环冗余校验(CRC)保护的闪存：被 EDC 纠正的单位故障不通过信号指示出来。该故障对安全目标的违背得到了 EDC 的预防，但未通过信号指示出来。如果 EDC 逻辑失效，该故障被 CRC 探测到，系统被关闭。只有当闪存中存在单位故障、EDC 逻辑失效、且 CRC 校验和监控失效时，才能发生对安全目标的违背（n=3）。

示例 2：三个电阻串联以克服短路情况下的单点故障问题，因为需要三个独立的短路才会违背安全目标（n=3），每个独立电阻的短路可被视为安全故障。

8.1.8 故障分类及故障类别贡献率计算的流程图

硬件要素的失效模式可按照GB/T XXXXX-5图B.1展示的、并使用GB/T XXXXX-5图B.2描述的流程图进行分类。图9展示了考虑基础失效率及不同失效模式（残余对潜伏）覆盖率的多种失效率计算。

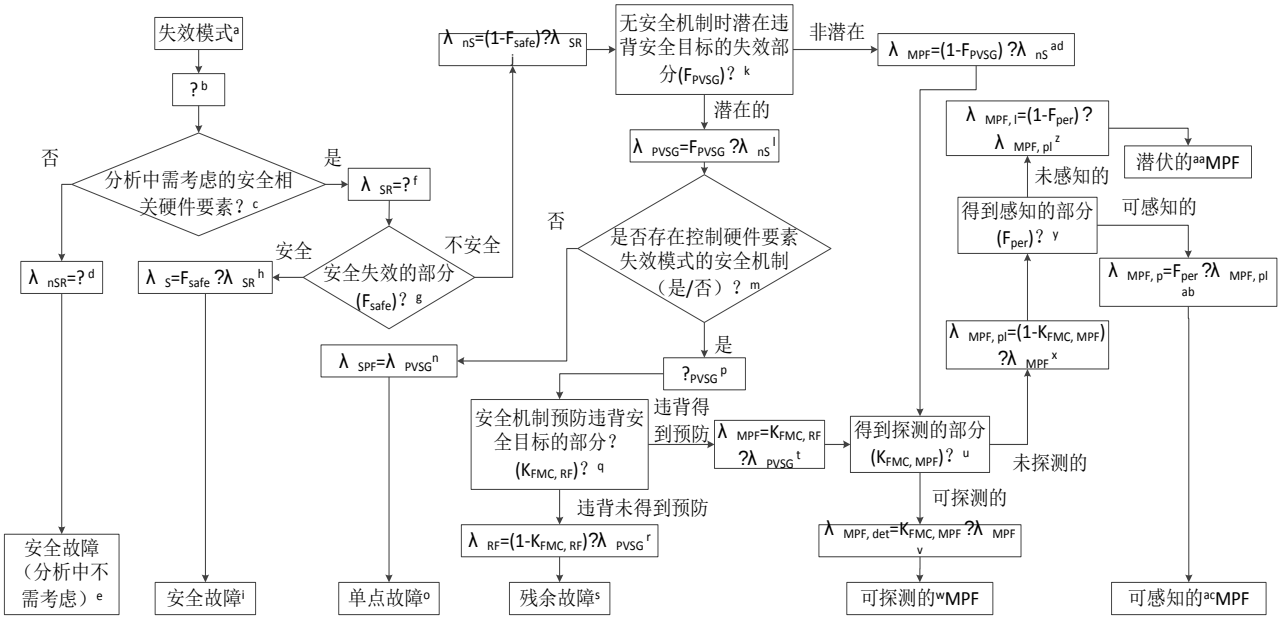


图 9 失效种类的划分和相应失效率的计算

[a]待分析的失效模式。

[b]λ 是与所分析失效模式相关的失效率。

[c]如果被分析的硬件要素的任何失效模式与安全相关，那么该硬件要素与安全相关。

[d]λ_{nSR}是非安全相关的失效率。如果所分析的硬件要素的全部失效模式与安全无关，那么$\lambda_{nSR}=\lambda$ 。

[e]与安全无关的故障是安全故障，且不包含在单点故障度量或潜伏故障度量中。

[f]λ_{SR}是安全相关失效率，在单点故障度量和潜伏故障度量中进行考虑。

[g]F_{safe}是该失效模式中的安全故障部分。安全故障不会显著导致安全目标的违背。对于复杂硬件要素（例如：微控制器），很难给出精确的比例。在此情况下，可假定保守的F_{safe}为0.5（即50%）。

[h]λ_{S}是安全故障的失效率，它等于$\lambda_{SR} x F_{safe}$。

[i]λ_{S}将增加安全故障的总体比率。

[j]λ_{nS}是不安全失效率，包括单点故障、残余故障和多点故障(n=2)。它等于$(1-F_{safe}) x \lambda_{SR}$。

[k] F_{PVSG}是λ_{nS}中潜在直接违背安全目标的部分（不考虑任何可能存在的防止其发生的安全机制）。

[l]λ_{PVSG}是潜在直接违背安全目标的故障的失效率（不考虑任何可能存在的防止其发生的安全机制）。它等于$F_{PVSG} \times \lambda_{nS}$。

[m]判断所考虑的引发失效模式的故障是否为单点故障。若没有实施安全机制以预防所考虑的硬件要素的任何违背安全目标的故障，则这些故障是单点故障。

[n]λ_{SPF}是单点故障失效率。如果控制所考虑的硬件要素失效率的安全机制一个都不存在，则全部的λ_{PVSG}都是单点故障。

[o]λ_{SPF}将增加单点故障的总体比率。

[p]对于所考虑的硬件要素，如果存在至少一个安全机制预防其至少一种违背安全目标的失效，那么导致该失效的故障不是单点故障。在后续过程中，λ_{PVSG}被分为残余故障和可探测的、可感知的及潜伏多点故障。

[q]安全机制预防了λ_{PVSG}中的哪部分对安全目标的违背？该部分等于针对残余故障的失效模式覆盖率[也可参见GB/T XXXXX-5附录E（硬件架构度量:“单点故障度量”和“潜伏故障度量”的计算示例）]。$K_{FMC, RF}$是针对残余故障的失效模式覆盖率的缩写。

[r]λ_{RF}是残余故障失效率。$\lambda_{RF}=(1-K_{FMC, RF}) \times \lambda_{PVSG}$。

[s]λ_{RF}增加了残余故障的总体比率。

[t]λ_{MPF}是多点故障失效率。$\lambda_{MPF}=K_{FMC, RF} \times \lambda_{PVSG}$。

u识别可探测故障和不可探测故障。$K_{FMC, MPF}$是针对多点故障的失效模式覆盖率。

注：多点故障存在两个来源：

——可导致安全目标违背，但存在安全机制预防的故障；

——本身不能导致安全目标违背，但会促成多点失效局面的故障。

针对多点故障的失效模式覆盖率可以随多点故障的来源不同而变化。

[v]$\lambda_{MPF, det}$是可探测的多点故障失效率。$\lambda_{MPF, det}=\lambda_{MPF} \times K_{FMC, MPF}$。

[w]$\lambda_{MPF, det}$增加了可探测的多点故障总体比率。

[x]$\lambda_{MPF, pl}$是可感知的或潜伏的多点故障失效率。

[y] F_{per}是$\lambda_{MPF, pl}$中被驾驶员感知的部分。

[z]$\lambda_{MPF, l}$是潜伏多点故障失效率。

[aa]$\lambda_{MPF, l}$增加了潜伏多点故障的总体比率。

[ab]$\lambda_{MPF, p}$是可感知的多点故障失效率。

[ac]$\lambda_{MPF, p}$增加了可感知的多点故障的总体比率。

[ad]λ_{MPF}是多点故障失效率。

8.1.9 如何考虑与基于软件的随机硬件失效安全机制有关的多点故障失效率

虽然软件和硬件的系统性故障在GB/T XXXXX中未被量化，对于支持运行基于软件的随机硬件失效安全机制的硬件资源，可计算其随机硬件失效的失效率。

如果那些硬件资源同时用于支持具有直接违背安全目标潜在可能的功能，那么选取反映此情况的失效模式并考虑潜在的相关失效。

8.2 残余失效率和局部单点故障度量评估的示例

8.2.1 总则

本示例阐述了一种评估传感器残余失效率$\lambda_{RF,Sensor}$、单点失效率λ_{SPF}和单点故障度量$M_{SPFM,Sensor}$本地化版本的方法。在此示例中，测量相同物理量并具有已知误差的两个传感器，一个传感器与另一个传感器的值相比较。应用功能使用传感器A_Master的值，另一个传感器A_Checker的值仅用于确认传感器A_Master的值。

此监控在GB/T XXXXX-5附录D中得到了引用，作为“传感器合理性检查”或“输入比较/投票”。

仅传感器A_Master的故障在本示例中得到了分类和评估，此处不针对传感器A_Checker的故障。

因传感器A_Master有一个已定义的安全机制，将存在的有违背安全目标潜在可能且未被控制的（即：不预防对安全目标的违背）全部故障定义为残余故障。单点失效率λ_{SPF}（按照定义）等于零。

8.2.2 传感器 A_Maste 的技术安全要求

图10中展示了传感器A_Master的安全运行边界，并作为此示例的前提（即：此处不讨论如何从安全目标导出）。它可用下述条款表示：

即

$$\mu_{SafRel,A,min} = Maximum(C_{PVSG}\ ;\ v\ \times\ (1 + a))$$

此处

C_{PVSG}是常值；

$\mu_{SafRel,A,min}$是传感器A_Master的安全下边界；

v是待测量的物理值；

a是常值。

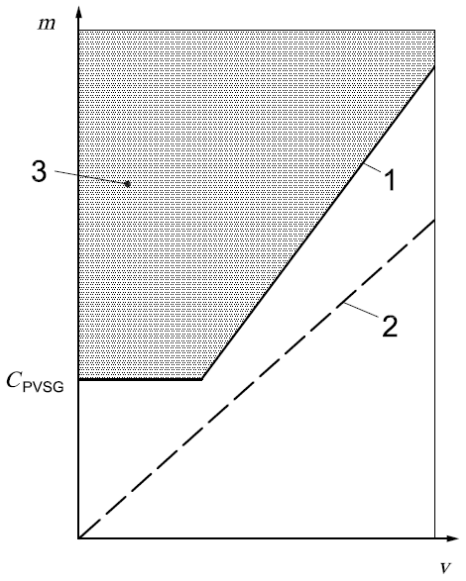
传感器的安全相关失效在如下条件时发生

$$m_{A,Master} \geq \mu_{SafRel,A,min}$$

此处

$m_{A,Master}$是传感器A_Master的返回值

安全要求是在故障容错时间间隔T_{SenA}内探测并控制传感器A_Master的安全相关失效。



略语表：

1 传感器A_Master的安全相关下边界$\mu_{SafRel,A,min}$

2带有0偏差的理想传感器返回值（作为参考）

3具有违背安全目标潜在可能的故障

图10 传感器A_Master的安全运行边界

图10中，x轴是需要测量的真实物理值v，y轴是传感器A_Master的返回值$m_{A,Master}$。虚线表示作为参考的理想传感器（即：具有0偏差的传感器）返回值。实线表示$\mu_{SafRel,A,min}$。如果传感器A_Master的返回值$m_{A,Master}$在实线上或高于实线，可能发生安全目标的违背。

8.2.3 安全机制的描述

安全机制的要素是传感器A_Checker和监控硬件，其包含带有嵌入式软件的微控制器。在小于容错时间T_{SenA}的周期内，软件周期性的对比两个传感器的值。通过以下伪码完成评估。

$\Delta_A = m_{A,Master} - m_{A,Checker}$

如果$\Delta_A \geq \Delta_{Max}$，那么失效为真

如果失效为真，那么进入安全状态

此处

$m_{A,Master}$是传感器A_Master的返回值；

$m_{A,Checker}$是传感器A_Checker的返回值；

Δ_{Max}是预先定义的作为通过/不通过准则的恒定最大门限值。

假设传感器有如下已知偏差：

$m_{A,Master} = v +/- c_{A,Master}$

$m_{A,Checker} = v +/- c_{A,Checker}$

此处

$m_{A,Master}$是传感器A_Master的返回值；

$m_{A,\ Checker}$是传感器A_Checker的返回值；

$c_{A,\ Master}$是代表传感器A_Master偏差的常值；

$c_{A,\ Checker}$是代表传感器A_Checker偏差的常值；

v是待测量的物理值。

选取量值Δ_{Max}以探测出传感器A_Master可能违背安全目标的失效。为了避免错误的失效探测，选择Δ_{Max}时考虑每个传感器的偏差及其它偏差的汇总$c_{A,\ other}$，例如：在不同时间采样的影响：

$$\Delta_{Max} \geq c_{A,\ Master} + c_{A,\ Checker} + c_{A,\ other}$$

用这种方法，探测不到失效的最严重情况是：

$$\mu_{A,\ Mater,\ wc} = m_{A,\ Checker} + \Delta_{Max}$$

$$= v + c_{A,\ Checker} + \Delta_{Max}$$

此处

$\mu_{A,\ Mater,\ wc}$是最差情况的探测门限，即：传感器A_Master探测到的、不作为失效的最大$m_{A,\ Master}$值；

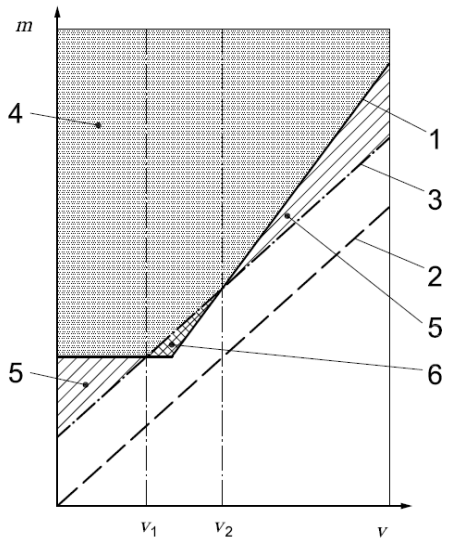
$m_{A,\ Checker}$是传感器A_Checker的返回值；

Δ_{Max}是预先定义的作为通过/不通过准则的恒定最大门限值。

v是待测量的物理值。

将高于$\mu_{A,\ Mater,\ wc}$的每个$m_{A,\ Master}$值定义为传感器失效。

取决于偏差值，不同的探测场景是可能的。图11和图12给出了两个可视化示例。



略语表：

1 传感器A_Master的安全相关下边界$\mu_{SafRel,A,min}$

2 带有0偏差的理想传感器返回值（作为参考）

3 最差情况的探测门限$\mu_{A,\ Mater,\ wc}$

4 探测到的双点故障

5 探测到的不会违背安全目标的故障

6 残余故障

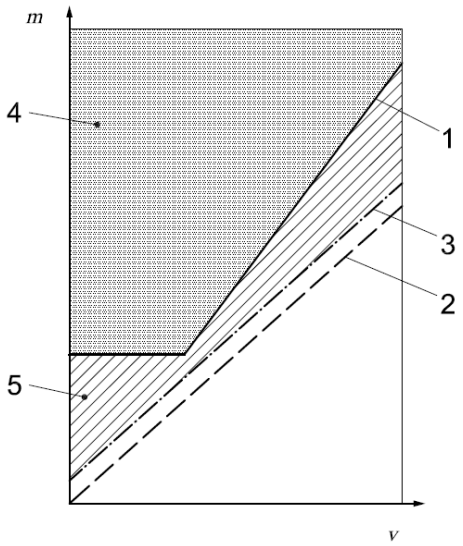
图11 最差情况探测门限（过高）的示例1

图11中箭头指示出三个区域。

区域5 – “探测到的不会违背安全目标的故障”是因其高于最差情况探测门限$\mu_{A,\,Mater,\,wc}$，而被安全机制探测出的故障，但因其低于安全相关下边界$\mu_{SafRel,A,min}$，其自身不会导致安全目标的违背。

区域4 –“探测到的双点故障”是可导致安全目标违背的故障，但其被安全机制探测并减轻。这些故障同时高于最差情况探测门限$\mu_{A,\,Mater,\,wc}$和安全相关下边界$\mu_{SafRel,A,min}$。这些故障的双点属性意味着需要安全机制和传感器同时失效才会导致对安全目标的潜在违背。

区域6 – “残余故障”未被安全机制探测且能直接导致安全目标的违背。对应$v\in[v_1, v_2]$的$\mu_{SafRel,A,min}<\mu_{A,\,Mater,\,wc}$区域，位于最差情况探测门限$\mu_{A,\,Mater,\,wc}$以下，但高于安全相关下边界$\mu_{SafRel,A,min}$。



略语表：

1 传感器A_Master的安全相关下边界$\mu_{SafRel,A,min}$

2带有0偏差的理想传感器返回值（作为参考）

3最差情况的探测门限$\mu_{A,\,Mater,\,wc}$

4探测到的双点故障

5探测到的不会违背安全目标的故障

图12 最差情况探测门限（$M_{SPFM,\,Sensor}$=100%）的示例2

在图12的情况，最差情况探测门限$\mu_{A,\,Mater,\,wc}$始终小于安全相关下边界$\mu_{SafRel,A,min}$。在此情况下，残余失效率为0，且传感器的局部单点故障度量$M_{SPFM,\,Sensor}$等于100%。

8.2.4 对图 11 所述示例 1 的评估

8.2.4.1 总则

在图11的情况，当传感器A_Master的最差情况探测门限$\mu_{A,\ Mater,\ wc}$高于安全相关下边界$\mu_{SafRel,A,min}$时，存在条件：

对于$v\in[v_1, v_2]$：$\mu_{SafRel,A,min}\leq\mu_{A,\ Mater,\ wc}$

为确定在这些条件下的残余失效率$\lambda_{RF,\ Sensor}$和$M_{SPFM,\ Sensor}$，进一步的分析是必要的。以下是分析示例。在GB/T XXXXX-5附录D中，下述失效模式表述为：

表2 传感器失效模式示例

要素	参考表格	60%、90%、99%诊断覆盖率对应的、所需分析的失效模式		
		低(60%)	中(90%)	高(99%)
总体要素				
传感器，含信号开关	**D.11**	无通用的故障模型。必要的细节分析。 应覆盖的典型的失效模式包括： • 超出范围 • 在范围之内卡滞	无通用的故障模型。必要的细节分析。 应覆盖的典型的失效模式包括： • 超出范围 • 偏移 • 在范围之内卡滞	无通用的故障模型。必要的细节分析。 应覆盖的典型的失效模式包括： • 超出范围 • 偏移 • 在范围之内卡滞 • 振荡

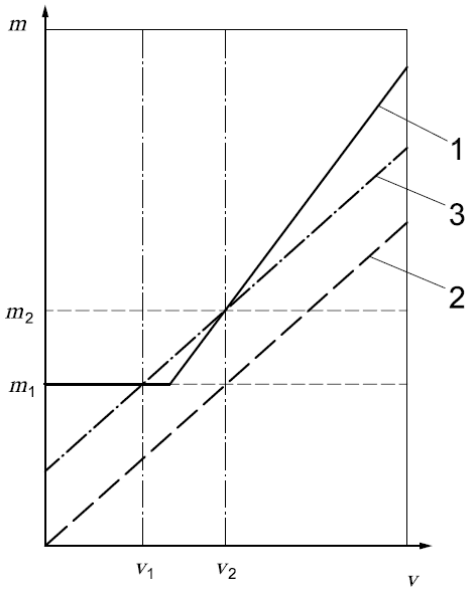
在本示例中，仅评估卡滞在恒定值m（范围内）。对于完整的传感器残余失效率及$M_{SPFM,\ Sensor}$的评估，全部其它失效模式都需要评估。

分析中，我们区分传感器三种不同的卡滞故障场景（参加图13）：

1) 传感器卡滞在值$m>m_2$；

2) 传感器卡滞在值$m<m_1$；及

3) 传感器卡滞在m_1和m_2之间的值m；



略语表：

1 传感器A_Master的安全相关下边界 $\mu_{SafRel,A,min}$

2 带有0偏差的理想传感器返回值（作为参考）

3 最差情况的探测门限 $\mu_{A,\ Mater,\ wc}$

图13 卡滞故障场景

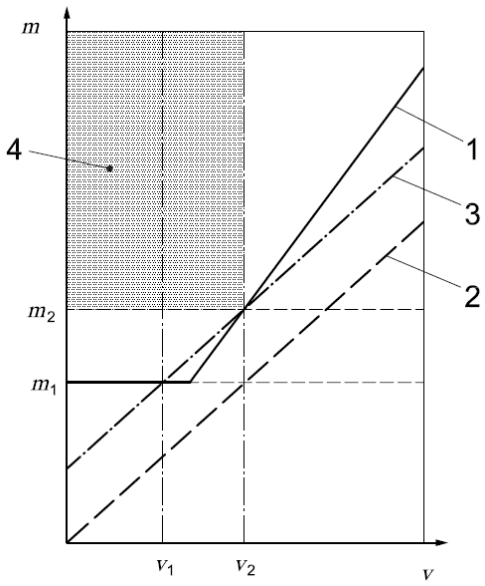
传感器卡滞故障在系统层面的影响取决于当前物理值v，例如：对于物理值$v \le v_2$，卡滞在m_2的故障有违背安全目标的潜在可能。对于值$v > v_2$，此故障没有违背安全目标的潜在可能。在以下分析中，考虑探测门限、物理值v及其分布概率，对故障成为残余故障的可能概率p_{RF}进行分析。

8.2.4.2 案例 1：传感器卡滞在值 m>m2 的故障

如果$v \le v_2$，故障有违背安全目标的潜在可能（参见图14）。然而，传感器的偏差始终高于最差情况的探测门限$\mu_{A,\ Mater,\ wc}$，所以安全相关的传感器失效及时得到了探测和控制。每个故障都是可探测的双点故障。在$v \le v_2$的情况下，残余故障率p_{RF}等于0。

如果$v > v_2$，故障不总是具有违背安全目标的潜在可能（参见图15）。如果该故障具有违背安全目标的潜在可能（图15区域6），它将高于最差情况探测门限并得到及时探测。注意，即使$v > v_2$时，某些故障（图15区域4和5）不会导致安全目标的违背，但它们不能被视为安全故障，因为当$v \le v_2$时，它们有违背安全目标的潜在可能。这些故障中的一部分位于最差情况探测门限以上并得到探测（图15区域4）。在$v > v_2$的情况下，残余故障率p_{RF}等于0。

如果$v \le v_2$，卡滞在$m > m_2$的故障具有违背安全目标的潜在可能，因而它们不能被视为安全故障。因所有的故障在其导致违背安全目标前，都得到了探测和控制，它们是可探测的双点故障；所以，对于卡滞在$m > m_2$的故障的残余故障率$p_{RF_stuck@m>m2}$等于0。



略语表：

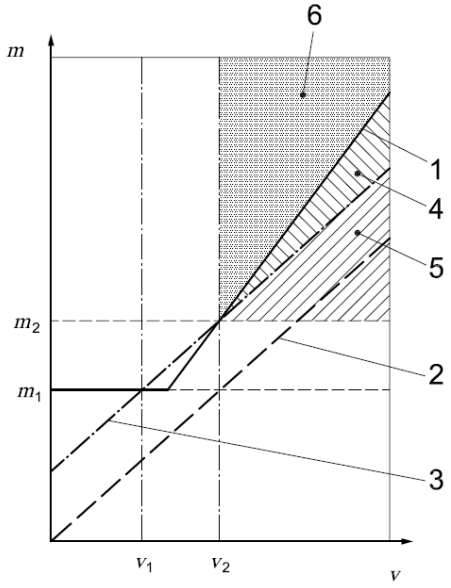
1 传感器A_Master的安全相关下边界$\mu_{SafRel,A,min}$

2 带有0偏差的理想传感器返回值（作为参考）

3 最差情况的探测门限$\mu_{A,Mater,wc}$

4 可探测的双点故障

图14 对卡滞在$m>m_2$且$v\leq v_2$的故障分类



略语表：

1 传感器A_Master的安全相关下边界$\mu_{SafRel,A,min}$

2 带有0偏差的理想传感器返回值（作为参考）

3 最差情况的探测门限$\mu_{A,Mater,wc}$

4 不具有违背安全目标潜在可能的可探测故障

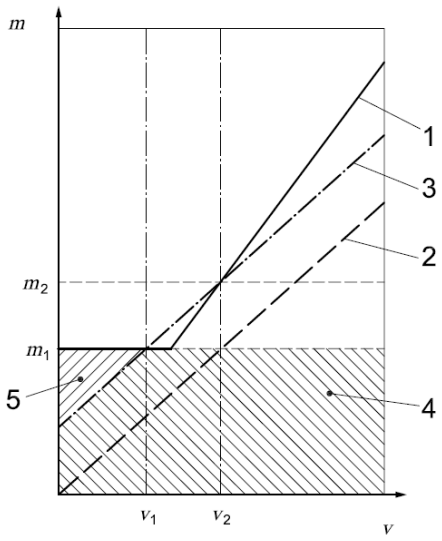
5 不具有违背安全目标潜在可能的不可探测故障

6 可探测的双点故障

图15 对卡滞在$m>m_2$且$v>v_2$的故障分类

8.2.4.3 案例 2：传感器卡滞在值 m<m1 的故障

图16中可见卡滞在$m<m_1$的故障。因在物理值v的全范围内，这些故障始终位于最差情况探测门限以下，不能导致安全相关的失效，它们是安全故障。所以，对于物理值v全范围内的残余故障，其故障率$p_{RF_stuck@m<m1}$等于0。



略语表：

1 传感器A_Master的安全相关下边界$\mu_{SafRel,A,min}$

2 带有0偏差的理想传感器返回值（作为参考）

3 最差情况的探测门限$\mu_{A,Mater,wc}$

4 不可探测的安全故障

5 可探测的安全故障

图16 对卡滞在$m<m_1$的故障分类

8.2.4.4 案例 3：传感器卡滞在值 m∈[m1, m2]的故障

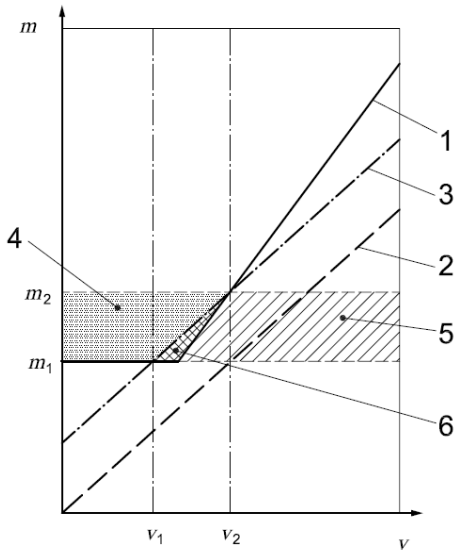
对卡滞在$m\in[m_1, m_2]$的故障，其违背安全目标的潜在可能和对其的探测取决于当前物理值v（参见图17），即：违背安全目标的概率取决于故障发生时v的当前值。针对故障发生时v的三个不同区间，对卡滞型残余故障失效率$p_{RF_stuck@m\in[m1,m2]}$进行评估。

——$v<v_1$；

——$v_1 \le v \le v_2$；

——$v > v_2$。

针对其中的每个条件，对残余故障率进行单独评估。最终残余故障率的计算，使用这三个故障率的值。



略语表：

1 传感器A_Master的安全相关下边界$\mu_{SafRel,A,min}$

2 带有0偏差的理想传感器返回值（作为参考）

3 最差情况的探测门限$\mu_{A,Mater,wc}$

4 可探测的双点故障

5 不会违背安全目标且不可探测的故障

6 残余故障

图17 对卡滞在$m\in[m_1, m_2]$的故障分类

取决于v的当前值，故障可以是可探测的双点故障（区域4）、残余故障（区域6）或不具有违背安全目标潜在可能的故障（区域5）。

$$p_{RF,stuck@m\in[m1,m2]} = p_{RF,stuck@m\in[m1,m2],v<v1}\cdot p_{v<v1} + p_{RF,stuck@m\in[m1,m2],v1\le v\le v2}\cdot p_{v1\le v\le v2} + p_{RF,stuck@m\in[m1,m2],v>v2}\cdot p_{v>v2}$$

此处

$p_{RF,stuck@m\in[m1,m2]}$是卡滞在值m（$m\in[m_1, m_2]$）并表现为残余故障的传感器故障率；

$p_{RF,stuck@m\in[m1,m2],v<v1}$是当故障发生时$v<v_1$、卡滞在值m（$m\in[m_1, m_2]$）并且表现为残余故障的传感器故障率；

$p_{v<v1}$是当故障发生时$v<v_1$的概率；

$p_{RF,stuck@m\in[m1,m2],v1\le v\le v2}$是当故障发生时$v1 \le v \le v2$、卡滞在值m（$m\in[m_1, m_2]$）并且表现为残余故障的传感器故障率；

$p_{v1\le v\le v2}$是当故障发生时$v1 \le v \le v2$的概率；

$p_{RF,\ stuck@\ m\in[m1,m2],\ v>v2}$是当故障发生时$v>v2$、卡滞在值$m$（$m\in[m_1, m_2]$）并且表现为残余故障的传感器故障率；

$p_{v>v2}$是当故障发生时$v>v2$的概率；

$p_{v<v1} + p_{v1\le v\le v2} + p_{v>v2}=1$

如果$v<v_1$，卡滞故障具有违背安全目标的潜在可能，但得到及时的探测，残余故障率$p_{RF,\ stuck@m\in[m1,\ m2],\ v<v1}$是0。

如果$v>v2$，卡滞故障不具有违背安全目标的潜在可能，但未得到探测。因为v值或早或晚介于v_1和v_2之间，则$p_{RF,\ stuck@\ m\in[m1,\ m2],\ v>v2}=p_{RF,\ stuck@\ m\in[m1,m2],\ v1\le v\le v2}$。

如果$v1\le v\le v2$，残余故障率$p_{RF,\ stuck@m\in[m1,\ m2],\ v1\le v\le v2}$不为0。

对保持在残余故障区域内足够长的时间并导致潜在违背安全目标的概率的确定并非没有意义。它可基于以下参数：

——物理值v及其概率分布的动态表现，例如：温度值更倾向于静态信号，而使用中的电机角度位置更倾向于动态信号；

——值v（$v\in [v1, v2]$）的概率分布；

监控软件的响应时间，例如：因过滤次数导致的响应时间。在示例中，单一事件$\Delta_A\ge\Delta_{Max}$足够用于探测传感器的失效并切换到安全状态。然而，作为通用实践，需实施一个错误计数器，进而必须有多于一次的事件以便评估传感器的失效并切换到安全状态。错误计数器的恢复，例如：当探测到非安全相关的事件（此示例中，将是$\Delta_A<\Delta_{Max}$）重置错误计数器，可对监控软件的探测能力产生显著影响，将大大降低其探测能力；

安全相关传感器导致潜在违背安全目标所必须的测量偏差次数。同时，也可能关注，必须介于安全相关传感器两个测量偏差内的有效测量次数，以便不再违背安全目标。

如果不具备每个影响参数的准确细节，使用专家判断和工程实践（例如：对未知的概率分布使用一个等效分布）得出保守估计是合理的。

基于已评估的不同概率$p_{RF,\ stuck@m>m2}$、$p_{RF,\ stuck@\ m<m1}$和$p_{RF,\ stuck@\ m\in[m1,\ m2]}$，可计算传感器残余卡滞故障率$p_{RF,\ stuck@\ m}$：

$$p_{RF,\ stuck@\ m}=p_{RF,\ stuck@\ m<m1}\times p_{m<m1} + p_{RF,\ stuck@\ m\in[m1,m2]}\times p_{m1\le m\le m2} + p_{RF,\ stuck@m>m2}\times p_{m>m2}$$

此处

$p_{m<m1}$是卡滞在$m<m_1$的故障率

$p_{m1\le m\le m2}$是卡滞在$m_1\le m\le m_2$的故障率

$p_{m>m2}$是卡滞在$m>m_2$的故障率

$p_{m<m1}+p_{m1\le m\le m2}+p_{m>m2}=1$

8.2.4.5 最终残余失效率评估

如果对每个相关失效模式FM_i按照上述方法进行评估，表现为残余故障的传感器总体故障率$p_{RF,\ Sensor}$可按如下进行计算：

$$p_{RF,Sensor} = \sum_i p_{FM,i} \times p_{RF,FM,i}$$

此处

$p_{FM,i}$是失效模式FM_i的概率

$p_{RF,FM,i}$是失效模式FM_i表现为残余故障的概率

$$\sum_i p_{FM,i} = 1$$

伴随此概率，残余失效率$\lambda_{RF,Sensor}$可被评估为

$$\lambda_{RF,Sensor} = p_{RF,Sensor} \times \lambda_{Sensor}$$

由此得出$M_{SPFM,Sensor}$

$$M_{SPFM,Sensor} = 1 - \lambda_{RF,Sensor} / \lambda_{Sensor} = 1 - p_{RF,Sensor}$$

8.2.4.6 SPFMSensor 的提升

降低传感器残余失效率的有效方法是减小Δ_{Max}的值。在下述条件中，减小Δ_{Max}不会显著增加误探测：

□ □公差的概率分布可能显示预估的最差情况是极其不可能的。因此，误报警概率足够低而可接受。

——系统的重新设计可得到改善的公差值。

注意，在此示例中，仅评估了传感器的故障，而非传感器存在路径中发生的故障。对于可导致两个传感器同时功能异常或可同时篡改两个传感器值的共享硬件资源（例如：微控制器的数模转换器ADC）的功能异常，进行单独评估。此外，完成如GB/T XXXXX-9第7章（相关失效分析）中给出的相关失效分析。

8.3 关于硬件的进一步解释

8.3.1 在 GB/T XXXXX 的应用范畴内如何处理微控制器

微控制器是车辆现代电子电气系统所必需的组件。可作为独立于环境的安全要素进行开发（参见第9章SEooC）。

对其复杂性的处理，是通过概念和产品开发阶段中，在恰当的抽象层面（即，从结构图到连线和布局层面）开展针对微控制器元器件/子器件的定性和定量联合分析实现的。

附录A是对如何在GB/T XXXXX范畴内处理微控制器的指导和非全面示例清单。

附录A描述了计算微控制器失效率的方法，包括如何考虑永久故障和瞬态故障。

它包含如下示例：

——相关失效分析；

——在微控制器设计过程中对系统性失效的避免；

——对微控制器安全机制的验证；及

——在系统层面对微控制器独立分析的考虑。

8.3.2 安全分析方法

附录B探讨了分析系统故障模式的技术方法，包括归纳分析和演绎分析，并包含故障树分析的示例。

8.3.3 随机硬件失效概率度量(PMHF)的计算中对暴露持续时间的考量

如GB/T XXXXX-5：9.4.2.3中描述的，定量分析为GB/T XXXXX-5：9.4.2.1要求的目标值得到满足提供了证据。如GB/T XXXXX-5：9.4.2.3中给出的，在双点故障情况下，此定量分析考虑暴露持续时间。

基于GB/T XXXXX-5：9.4.2.3中的注2，暴露持续时间开始于故障发生时。

它包括：

——每个安全机制相关的多点故障探测时间间隔，或当故障不被驾驶员所知（潜伏故障）时，车辆的生命周期；

——一次行程的最长时间（驾驶员被要求以安全方式停车的情况）；及

——出现警示到车辆在维修站得到修理的平均时间间隔（驾驶员被警示修理车辆的情况）。

以下示例展示了考虑暴露持续时间的可能方法。在此示例中，假定预期功能（任务块“m”）得到安全机制“sm”的监控。

随机硬件失效概率度量MPMHF的值（考虑安全机制先于任务块失效的条件概率），可使用下述公式进行计算：

$$M_{PMHF} = \frac{\lambda_{m,RF} \times T_{Lifetime} + \lambda_{m,DPF} \times T_{Lifetime} \times 0.5 \times \left(\lambda_{sm,DPF,latent} \times T_{Lifetime} + \lambda_{sm,DPF,detected} \times \tau_{SM}\right)}{T_{Lifetime}}$$

此处

M_{PMHF}是随机硬件失效概率度量(PMHF)的值

$\lambda_{m,RF}$是预期功能（任务块“m”）的残余失效率

$\lambda_{m,DPF}$是任务块“m”的双点失效率

$T_{Lifetime}$是车辆生命周期

$\lambda_{sm,DPF,latent}$是安全机制“sm”的潜伏双点失效率

$\lambda_{sm,DPF,detected}$是安全机制“sm”的可探测双点失效率

τ_{SM}是安全机制“sm”的多点故障探测时间间隔

如果$\lambda_{m,DPF} \times \lambda_{sm,DPF,detected} \times \tau_{SM}$（代表了在同一$\tau_{SM}$内，任务与相关安全机制同时失效的概率）非常小[例如：如果τ_{SM}是一次驾驶周期的量级（即使$\lambda_{m,DPF}=\lambda_{sm,latent}$=1000FIT，此示例中的贡献≤$10^{-12}$）]，其可被忽略，则公式简化为：

$$M_{PMHF} = \lambda_{RF} + 0.5 \times \lambda_{m,DPF} \times \lambda_{sm,DPF,latent} \times T_{Lifetime}$$

如果条件概率不适用（例如：失效的次序不相关），公式变化为：

$$M_{PMHF} = \lambda_{RF} + \lambda_{m,DPF} \times \lambda_{sm,DPF,latent} \times T_{Lifetime}$$

9 独立于环境的安全要素

9.1 脱离环境的安全要素的开发

汽车工业为不同的客户和不同的应用开发通用的要素。这些通用的产品是在不同的组织中独立开发出来的。在这种情况下，先做出关于需求以及设计的假定，这些假定包括了通过更高设计层级以及要素外部设计而得到的分配到要素的安全要求。

这样开发出来的要素可以当作是脱离环境的安全要素（SEooC）。SEooC是与安全相关的要素，它不是为了一个特定相关项而开发的。这意味着它不是在一个特定车辆环境中开发出来的。

SEooC可以是系统，系统阵列，子系统，软件组件，硬件组件或者零部件。SEooC的例子包括系统控制器，ECUs，微控制器，执行通信协议的软件，或者AUTOSAR软件组件。

SEooC不可以是一个相关项，因为相关项总是需要用于批量生产的整车环境。如果SEooC是一个系统，该系统不在这个环境中开发的，那么它就不是一个相关项。

SEooC与GB/T XXXXX-8第12章（软件组件的鉴定）和GB/T XXXXX-8的第13章（硬件组件的鉴定）所描述的经鉴定的组件的区别在于：

——SEooC是依照GB/T XXXXX基于假设开发的。若在集成SEooC过程中可以证实该SEooC的假设的有效性，则它可用于多个不同的相关项。

——软硬件组件的鉴定表述了按照GB/T XXXXX开发的相关项对于已有要素的使用。这些组件不是为了复用性设计的，也不是按照GB/T XXXXX开发的。

表3描述了不同软件要素的鉴定、脱离环境的安全要素和在用证明的预期使用。

在GB/T XXXXX第6章7.4.6中描述了表3中对于软件组件的分类。

表3 软件组件的分类

软件组件的分类	第6部分 在相关项的环境中	第8部分第12章 软件组件的鉴定	第6部分 作为脱离环境的安全要素	第8部分第14章 在用证明的论据
全新开发	适用	不适用	适用	不适用
带变更的复用	适用	不适用	适用	适用
不带变更的复用	不适用	适用	适用 （如果作为SEooC来开发）	适用
参见GB/T XXXXX-8, 14.4.4				

当开发一个SEooC时，可根据GB/T XXXXX第2章6.4.5.6对所应用的安全活动进行裁剪。这种对于SEooC开发的裁剪并不意味着安全生命周期中的任何一个步骤都可被省略。即使在SEooC的开发中某些步骤推迟了，它们也会在相关项开发中也完成的。

SEooC的ASIL能力标明了该SEooC遵从带指定ASIL的安全要求的能力。所以，它定义了应用于该SEooC开发的GB/T XXXXX的需求。

因此，SEooC是基于假设而开发的；假设了一个既定的功能以及包含外部接口的使用环境。这些假定是以一种阐述相关项父级的方式而建立的，故后期SEooC可在多个不同但却相似的相关项中使用。即使在SEooC的开发中某些步骤推迟了，它们也会在相关项开发中也完成的。

在实际相关项的环境中集成SEooC时，证实这些假设的有效性。

可由多个SEooC搭建一个相关项，而各SEooC之间的接口是互相直接连接的。这种情况下，对其中一个SEooC假设的有效性证实需要考虑到接口的SEooC。

如果在SEooC集成到相关项的过程中无法证实该SEooC在开发过程所做的假设的有效性，则需要对该SEooC或者该相关项进行GB/T XXXXX-8第8章（变更管理）中所描述的更改。

9.2 使用案例

9.2.1 概述

SEooC的开发包括对产品开发的相应阶段做出前提条件的假设，例如对于一个软件组件，它是软件架构设计的一部分，相应的阶段就是GB/T XXXXX-6第7章的子阶段。无须对所有的前提条件做假设，例如安全计划。

图18表示了假设与SEooC开发之间的关系。一个SEooC的开发可以从需求和设计的某个层级开始。需求或设计的前提条件的每一条信息都预定义为“假设”状态。

对SEooC需求（由假设的高级别需求和假设的该SEooC外部设计而派生出来的）的正确实施将在SEooC开发过程中得到验证。

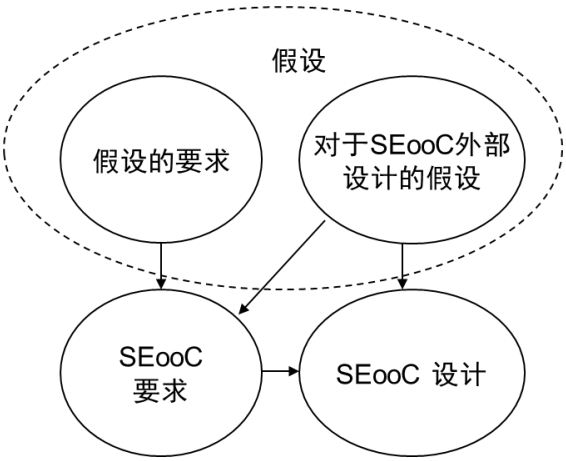


图18 假设与SEooC开发之间的关系

对 SEooC 需求（由假设的高级别需求和假设的该 SEooC 外部设计而派生出来的）的正确实施将在 SEooC 开发过程中得到验证。这些需求和假设的确认在相关项开发过程中进行确认。

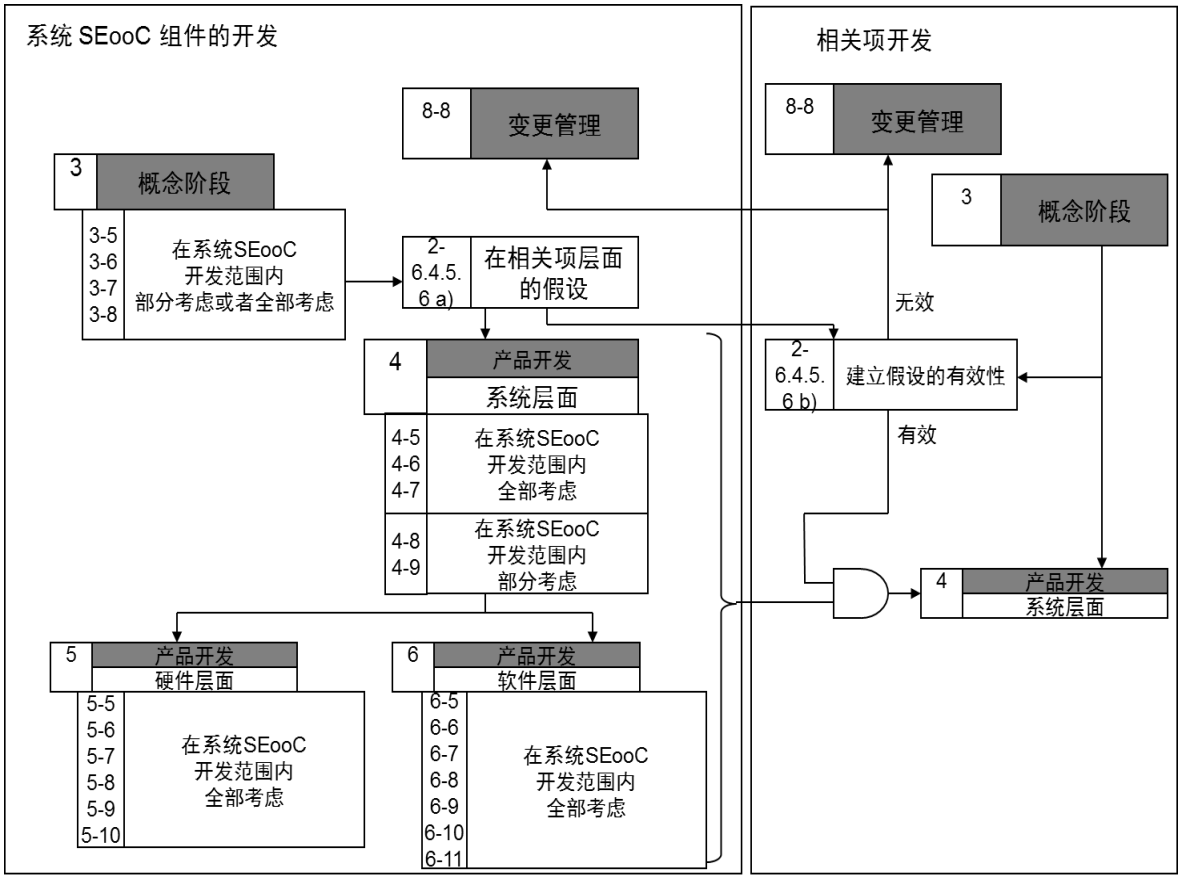
类似的，确认活动表明了一个已经开发好的SEooC，在任何层面，是符合其使用环境需求的。例如，当使用一个脱离环境而开发的软件组件，对软件规范的确认能证明其满足软件架构设计规范中的需求。在SEooC开发完成，并且相关项开发达到安全要素需求形成的阶段，可以产出该确认报告。

下面给出了SEooC的一些典型例子，即系统，硬件组件和软件组件。

9.2.2 开发一个作为 SEooC 的系统

本节用于说明一个可被不同整车制造商所集成的的新的E/E系统中，如何应用SEooC概念的裁剪工作。

为了说明，系统包含的功能不仅有在特定整车条件下的功能激活，还允许在适当的驾驶员要求下功能停止。过程流如图19所示。



注1：有必要依据SEooC的确切特性，对需求做额外的裁剪。

注2：依据SEooC的确切特性，第三部分和第四部分的一些需求可能不适用，所以只做了部分考虑。

注3：尽管不能给出所有的GB/T XXXXX章节，但这并不意味着它们是不适用的。

图19 SEooC系统开发

步骤1a - 对SEooC范围的定义

基于假设，SEooC的开发者定义SEooC的目的，功能以及外部接口。

对SEooC范围的这些假设，可举例如下：

——系统是针对总质量不超过 1800kg 的车辆而设计的。

——系统是针对四驱车辆而设计的。

——系统是针对不超过 32%的道路坡度而设计的。

——系统有对外部系统的接口，用于获取所需的车辆信息。

——功能性要求：

——系统在某些车辆状况下应驾驶员的请求而激活功能；

——系统应驾驶员的请求而解除功能；

步骤 1b - 对SEooC安全需求的假设

为了识别SEooC的技术安全要求，开发一个SEooC需要对与SEooC功能相关的相关项定义、相关项的安全目标、以及相应的功能安全需求做出假设。

对分配到SEooC的功能安全需求假设，可举例如下：

——系统在高车速时不会激活功能（ASIL x）。

——系统不会在没有检测到驾驶员请求的时候解除功能（ASIL y）。

为了达到所假设的安全目标，可定义针对背景的特定假设。

对于SEooC环境的假设，可举例如下：

——有一个外部源，可以提供带所需ASIL等级的信息，使得系统能检测到适当的车辆状况（ASIL x）。

——有一个外部源，可以提供关于驾驶员请求的信息，且该信息达到所需的 ASIL 等级（ASIL y）。

步骤 2 – SEooC 的开发

当技术安全要求由相关项假定的功能安全需求导出后，则SEooC遵循GB/T XXXXX的要求来开发。

步骤3 – 工作成果

在SEooC开发的最后阶段，可得到用于展示所导出的技术安全要求已全部得到满足的工作成果。然后，将工作成果中的所有必要信息，包括SEooC的安全要求以及对于环境的假设，提供给相关项的集成者。

步骤4 – SEooC集成到相关项

在相关项开发中，定义安全目标和功能安全要求。相关项的功能安全要求与SEooC所假设的功能安全要求是匹配的，以证明假设的有效性。

如果发生某个SEooC的假设不匹配，要开展从影响分析开始的变更管理活动，如GB/T XXXXX-8，第8章（变更管理）所述。可能的结果包括：

——根据安全目标的达成情况，差异是可接受的，没有行动要做；

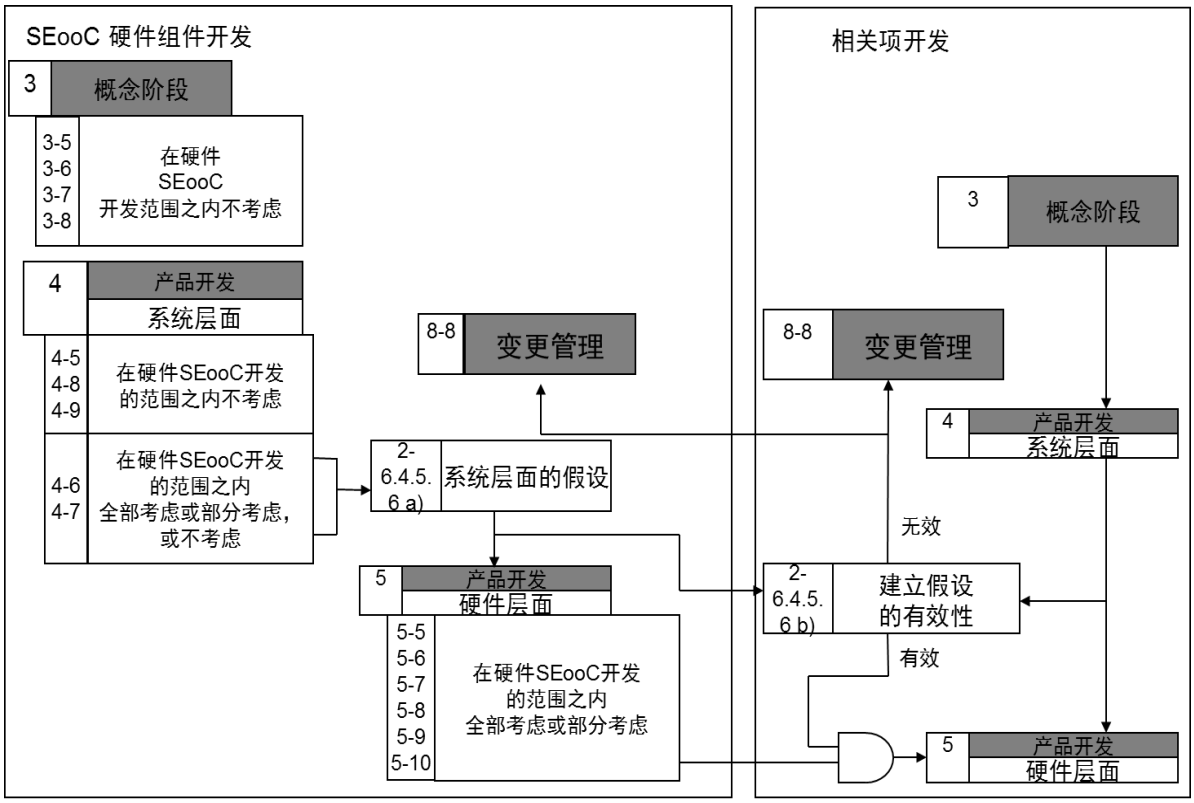
——差异影响到了安全目标的完成，需有一个相关项定义或者功能安全概念的必要的变更；

——差异影响到了安全目标，需有一个对于 SEooC 组件的变更（包括组件的一个可能的变更）。

9.2.3 开发一个作为脱离环境的安全要素的硬件组件

9.2.3.1 概述

本节使用附录A的微处理器（MCU）例子作为SEooC的硬件示例。流程图如图20所示。



注1:依据SEooC的确切性质，一些对需求的额外剪裁是必须的。例如根据随机硬件失效调整违反某个安全目标概率的目标值。

注2:依据SEooC的确切性质，第五部分的一些要求将不适用，因此只做了部分的考虑。

注3:尽管没有表明GB/T XXXXX的所有条款，这并不意味着它们是不适用的。

图20 SEooC硬件组件开发

9.2.3.2 步骤 1- 系统层面的假设

开发一个作为SEooC的微处理器（MCU）（如图20），首先是依据GB/T XXXXX-2, 6.4.5.6对系统层面属性和需求的假设。

基于一些参考应用的分析，本阶段可以分解成两个子步骤（1a和1b）。需求的假设要考虑硬件产品开发（GB/T XXXXX-5，表A.1）的前提条件。举例如下。

9.2.3.3 步骤 1a –对于技术安全要求的假设

下面是对于MCU示例的一些假设的技术安全要求的例子：

对于技术安全要求（步骤1a）的假设

a) 通过在硬件方面的安全机制来减轻 CPU 指令存储器的失效，至少满足分配到硬件元器件层面的单点故障度量（也可是“要求的诊断覆盖率”）的目标值（例如 90%）。

b) MCU 对于违反一个安全目标的总体可能性的贡献不能超过相关 ASIL 所允许可能性的 10%。

c) 复位时， MCU 实施一个已定义为所有 I/O 驱动输出到低状态的安全状态。

d) 任何与正在处理的功能相关的安全机制要在小于 10 ms（所分配到的部分故障允许时

间间隔）内完成。

e) 在安全相关的操作中没有用到 MCU 的调试接口。所以，调试逻辑中的任何故障都可认为是安全故障。

f) 有一个存储保护单元用以隔离不同 ASIL 的软件任务。

在这一步，建立起了ASIL能力。

9.2.3.4 步骤 1b –系统层级设计的假设

一些系统层级设计假设的例子，对 SEooC 的外部：

a) 系统将在 MCU 的供电上实施安全机制以检测过压失效模式和欠压失效模式。

b) 系统将在 MCU 外部实施一个窗口看门狗安全机制以检测 MCU 的时序失效或者程序顺序失效。

c) 将实施一个软件测试以检测 MCU(SM4)的 EDC 安全机制中的潜伏故障。

d) 在 Key-on 时会执行一个基于软件的测试（SM2）以确认对 CPU 程序顺序的逻辑监控（SM1）中不存在潜伏故障。

9.2.3.5 步骤 2– 硬件开发的执行

基于这些决定（假设的技术安全要求以及与SEooC外部设计相关的假设），根据GB/T XXXXX-5进行SEooC的开发（步骤2），并准备每一项适用的工作成果。例如，考虑对于SEooC的假设，包括在假设的技术安全要求中发现的任何FIT率目标，完成对于由随机硬件失效引起的安全目标违反的评估（参见GB/T XXXXX-5，9.5.1所写的工作成果）。基于SEooC的假设，根据GB/T XXXXX-9进行安全分析和MCU内部的关联失效分析。

对于A.3.5中的MCU示例，安全要求a) 是得到满足的，因为存储器的单点失效度量（99.8%永久失效和99.69%瞬态失效）是大于该硬件部件所分配到的90%目标。系统设计上的假设c)是由安全机制SM4来实施的。

9.2.3.6 步骤 3 – 工作成果

在MCU产品开发的最后（步骤3），将工作成果中的必要信息提供给系统集成者；这包括以下文档：假设的要求，与SEooC外部设计相关的假设以及适用的GB/T XXXXX工作成果（例如，针对由随机硬件失效而违反安全目标的可能性的报告）。

9.2.3.7 步骤 4–将 SEooC 集成到相关项

当作为SEooC开发的MCU在相关项硬件产品开发阶段的环境中予以考虑时，所有关于SEooC的假设，包括假设的技术安全要求和针对SEooC外部设计的假设，都需要建立有效性（步骤4）。有可能会出现SEooC假设和系统要求之间的不匹配。例如，相关项的开发者可以决定不实施假设的外部组件。由此，SEooC开发者所做的因随机硬件失效而导致的违反安全目标的评估就与相关项不一致了。

如果发生某个SEooC的假设不匹配，要开展从影响分析开始的变更管理活动，如GB/T XXXXX-8，第8章（变更管理）所述。可能的结果包括：

——根据安全目标的达成情况，差异是可接受的，没有行动要做。

——差异影响到了安全目标的完成，要有一个对相关项定义或者功能安全概念的必要的变更。

——差异影响到了安全目标的完成，要有一个对 SEooC 组件的变更（可能包括组件的一

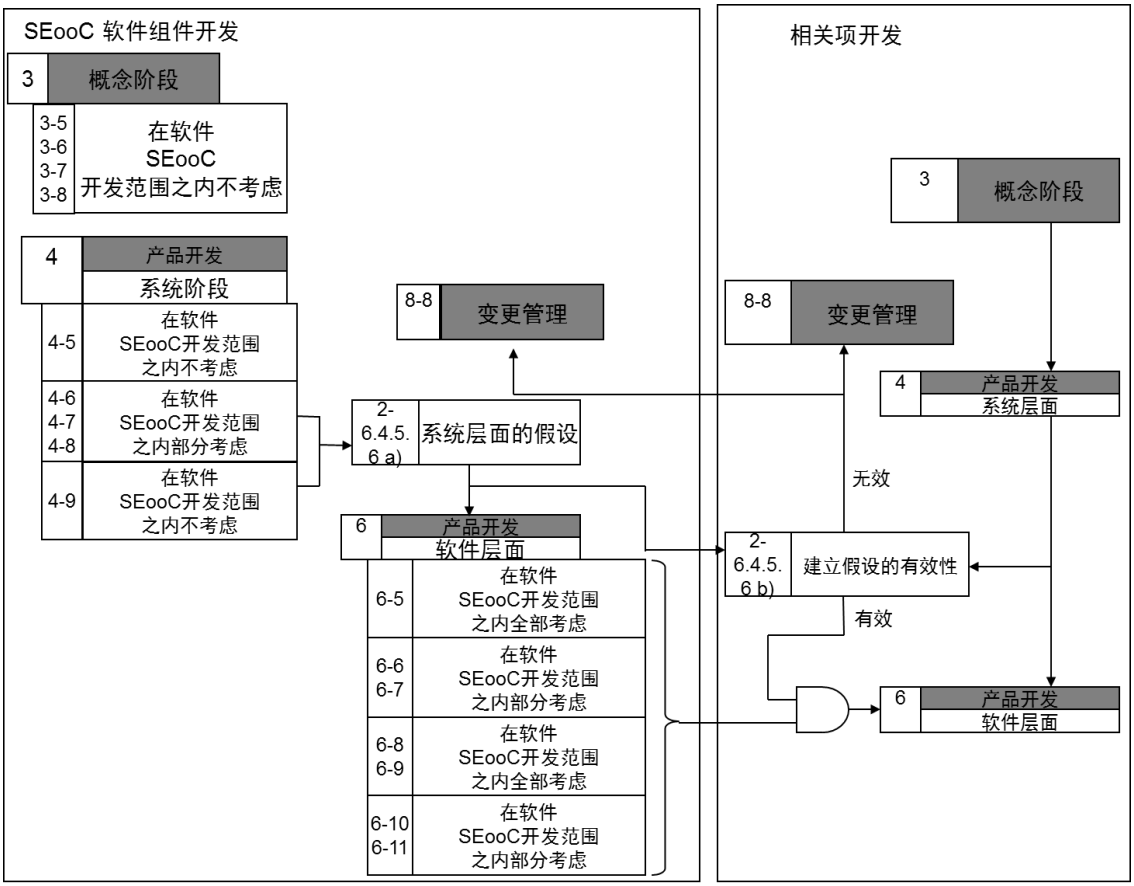
个变更）。

——差异影响到了安全目标，由此重新计算安全度量，但是这个重新计算的度量表明设计是满足系统目标的，所以无须更改。

9.2.4 开发一个作为脱离环境安全要素的软件组件

9.2.4.1 概述

本节说明了将SEooC概念应用于一个新的中/低层软件组件。流程图如图21所示。



注1：依据SEooC的确切性质，一些对额外的需求剪裁是必须的。

注2：依据SEooC的确切性质，第6部分的一些要求将不适用，因此只有部分的考虑可以去做。

注3：尽管没有表明GB/T XXXXX的所有条框，这并不意味着它们是不适用的。

图21 SEooC软件组件开发

9.2.4.2 步骤 1a- 对于作为 SEooC 软件组件的范围的假设

本步骤用于陈述关于软件组件的目的，范围，环境以及功能的相关假设。

这些假设的例子可包括：

——集成软件组件到一个给定的软件层级架构中。

——由软件组件引起的任何潜在干涉是在其环境中监测和处理的。

——软件组件提供下列功能：功能性软件要求清单。

9.2.4.3 步骤 1b –对于软件组件安全要求的假设

步骤1b用于假设潜在影响软件组件的更高层级的软件要求，以便导出软件安全需求。例如，如果假设一组给定的由软件组件计算得到的数据是具备高完整性（ASIL x），那么由此得到的分配到SEooC的软件安全要求可以是：

——软件组件对于下列输入数据检测任何数据损害：输入数据清单（ASIL x）；

——软件组件提示下列错误条件：错误条件清单（ASIL x）；

——返回一个默认值，带有一个检测到任何错误情况的故障状态（ASIL x）；

——软件组件返回经 CRC 编码的下列结果，以及返回一个状态（ASIL x）。

9.2.4.4 步骤 2–软件组件的开发

一旦针对软件组件的必要假设得到清晰的规定，根据SEooC的ASIL能力（在本示例中为ASIL x），按照GB/T XXXXX-6的要求，进行SEooC的开发。完成所有适用的工作成果，包括与假设的软件安全要求的验证相关的工作成果，以用于不同环境中的进一步集成。

9.2.4.5 步骤 3– 全新特定环境下的软件组件的集成

软件组件与其它软件组件在全新特定环境下进行集成之前，检查SEooC所做的全部假设在该环境下的有效性。这包括假设的带ASIL能力的软件安全要求，和所有关于软件组件的目的，边界，环境以及功能的假设（参见9.2.4.2和9.2.4.3）。

如果发生某个SEooC的假设与该新环境不匹配，要开展从影响分析开始的变更管理活动，如GB/T XXXXX-8，第8章（变更管理）所述。影响分析的可能结果包括：

——根据软件架构设计层面所适用的安全要求的达成情况，差异是可接受的，没有行动要做。

——差异影响到了软件架构设计层面所适用的安全要求的达成情况。针对特定情况，根据GB/T XXXXX-8，第 8 章（变更管理），执行对软件组件或者软件架构设计层面所适用的软件要求的变更。

注：当一个软件组件集成到特定的软件架构设计中导致了被分配到不同ASIL等级的软件安全相关要素的共存，那么需如GB/T XXXXX-9，第6章（要素共存准则）所描述的，满足要素共存的准则，或者作为选择的，低ASIL等级的要素升级到高ASIL等级。

10 在用证明的示例

10.1 总则

本章所描述的相关项及其要求是一个示例。这里给出了安全目标、其ASIL等级和后续要求，以说明GB/T XXXXX-8，第14章（在用证明）所定义的在用证明。本示例不反映GB/T XXXXX-8对于类似的现实案例的应用。

10.2 相关项定义和在用证明候选项的定义

整车制造商想在新车上集成一个新的功能。例如：实施该功能的相关项由一些传感器，一个ECU（含功能所需的完整软硬件），以及一个执行器组成。

整车制造商把功能的异常动作评定为ASIL C等级。相应的安全目标派生出一个分配到ECU的ASIL C等级的功能安全要求。

ECU的供应商提议沿用一个现有的ECU。

分析ECU之前使用和它将要在新应用中使用的区别。分析表明，需要通过更改标定数据来修改软件以实施新的功能，但是ECU的硬件可以沿用，不做改动。供应商想用ECU硬件的在用证明来代替符合GB/T XXXXX-5要求的说明。因此，该ECU的硬件就是在用证明的候选项。

10.3 变更分析

为了建立在用证明的可信度，供应商对在用证明的候选项开展变更分析。

该分析表明，从ECU的量产开始，没有引入对在用证明候选项的安全行为产生影响的变更。

此外，分析表明在用证明候选项的先前应用与预期应用之间的差异没有安全影响：

——候选项边界在规范限制之内；

——先前的集成环境需要相同的技术行为；及

——候选项边界在先前和未来的集成环境中是相同的原因及影响。

10.4 在用证明的目标价值

为了证实在用证明的有效性，供应商预估了在用证明候选项已经在现场使用的累计小时数。供应商也分析了售后期间任何与安全相关事件的现场数据，即，有关新的相关项中候选项的预期用途，已报告的任何潜在导致或促使违背安全目标或安全要求的事件。

基于搭载在用证明候选项的量产车辆的数量、车辆量产日期、及在该细分市场中车辆典型使用数据（每年驾驶小时数），对维修历史的持久性进行预测。

维修历史是基于搭载在用证明候选项的不同车辆的现场返修数据：

——保修索赔；

——现场缺陷分析；或

——从整车制造商处返回的缺陷零件。

在相关项硬件开发之初，这些分析表明现场没有发生安全相关的事件。预测的总体累计驾驶小时数小于ASIL C的在用证明的确定目标，但是满足GB/T XXXXX-8, 14.4.5.2.5定义的临时服务期限。

由此结论如下：

——ECU 硬件的临时在用证明可继续被相关项开发所采信。

——继续进行现场观察以获取一个确定的在用证明的状态（参见 GB/T XXXXX-8, 14.4.5.2.5 和 GB/T XXXXX-8, 14.4.5.2.6）。

11 关于 ASIL 的分解

11.1 ASIL 分解的目的

ASIL 分解的目的是针对系统性失效，应用冗余以满足安全目标。ASIL分解可得出冗余的要求及相应的分解后的ASIL等级，这些要求及其ASIL等级由充分独立的要素来实现。

11.2 ASIL 分解的描述

ASIL分解是指将冗余的安全要求分配到充分独立的相关项要素上。此处，冗余不一定是传统的模块化冗余（参见GB/T XXXXX-1, 1.94）。

示例:ECU的主处理器可被一个冗余的监控处理器所监控，即使监控处理器不能实现分配到ECU的功能要求，主处理器和监控处理器均能独立的启动已定义的安全状态。

ASIL分解仅对系统性失效有意义，也就是说，它是降低这些失效可能性的方法和途径。ASIL分解不会改变对硬件架构度量的评估要求，也不会改变对随机硬件失效导致违背安全目标的评估要求（参见GB/T XXXXX-9, 5.4.5）。

示例:对于ASIL B(D)分解的案例，在硬件架构度量评估中，不允许将ASIL D的目标分解成对于每个硬件要素的ASIL B目标。如GB/T XXXXX-5, 8.2所述，可以将目标值分配到硬件要素，但是这些目标是基于在相关项的整体硬件层面开始的分析而逐项分配的。安全目标的目标度量是应用到相关项层面的。

在这样一个分解后的架构里，只有当两个要素同时违背了分解后的安全要求，才会违背相关的安全目标。

在GB/T XXXXX-9，第5章（关于ASIL裁剪的要求分解）中描述了GB/T XXXXX中所允许的分解。

11.3 ASIL 分解的示例

11.3.1 总则

本章中描述的相关项及其要求均为示例。安全目标及其ASIL等级和后续要求，仅为了说明ASIL分解过程而设计的。本示例不能反映GB/T XXXXX在类似的现实示例上的应用情况。

11.3.2 相关项定义

以具有一个执行器的系统为示例，驾驶员通过使用仪表板上的开关来触发此执行器。执行器在车速为零时提供舒适功能，但是如果在超过15km/h时激活会导致危害。

相关项的初始架构如下：

——由一个专门的 ECU（本示例中称之为“AC ECU”）读取仪表板开关的输入，该 ECU 通过一个专门的电源线为执行器供电。

——搭载本相关项的车辆同时也带有一个能提供车速的 ECU（本示例中称之为“VS ECU”）。假定此 ECU 能按照 ASILC 的要求提供车速超过 15km/h 的信息。

11.3.3 危害分析和风险评估

本分析中考虑的危害性事件是在车速超过15km/h时激活执行器，无论此时是否存在驾驶员请求。

本危害事件的ASIL分级为ASIL C。

11.3.4 相关的安全目标

安全目标1：避免在车速超过15km/h时激活执行器：ASIL C

11.3.5 初步架构和安全概念

11.3.5.1 总则

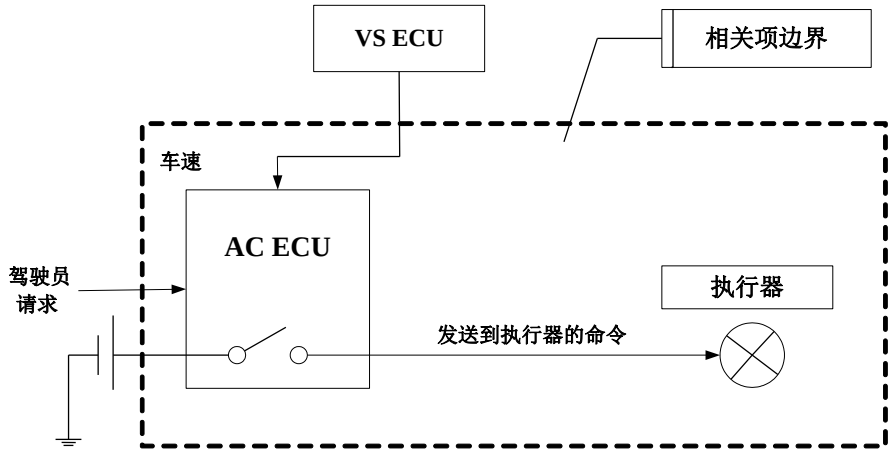


图 22 相关项边界

11.3.5.2 要素的目的（初始架构）

——动态 VS ECU 为 AC ECU 提供车速。

——AC ECU 监控驾驶员请求，检测车速是否小于或等于 15km/h，若是，则发送命令到执行器。

——执行器通电后就被激活。

11.3.6 功能安全概念

11.3.6.1 总则

本功能安全概念的示例仅用于说明 ASIL 分解，该示例并不完整，未包括全部的功能安全要求；

——要求 A1: VS ECU 发出准确的车速信息给 AC ECU。➔ ASIL C

——要求 A2: 当车速超过 15km/h 时，AC ECU 不能给执行器供电。➔ ASIL C

——要求 A3: 执行器只有在得到 AC ECU 的供电之后才能被激活。➔ASIL C

11.3.6.2 演化后的相关项安全概念

开发者可以选择引入一个冗余的要素，在这里是一个安全开关，如图23所示。通过引入这个冗余的要素， AC ECU可根据ASIL分解的结果，按照等同于或者低于ASIL C的一个ASIL等级来进行开发。

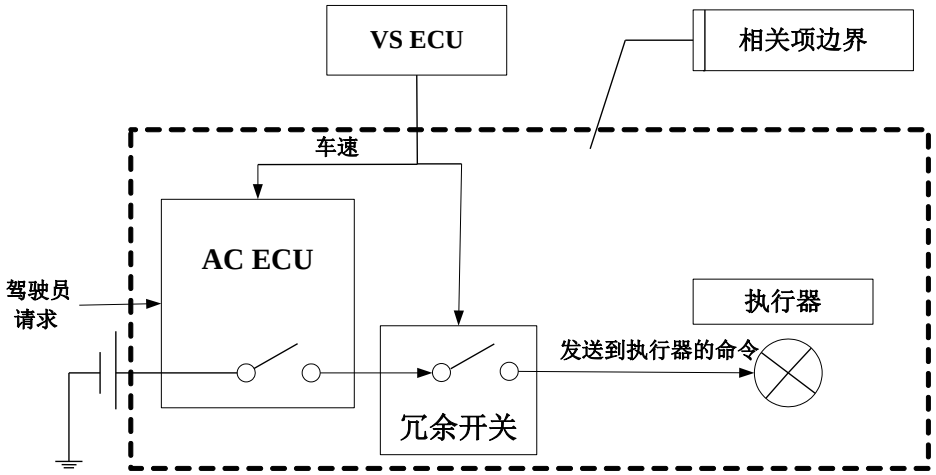


图 23 相关项设计的第二次迭代

这些要素的目标（演化后的架构）：

——VS ECU 控制单元提供车速给 AC ECU。

——AC ECU 监控驾驶员请求，检测车速是否小于或等于 15km/h, 若是，则发送命令到执行器。

——冗余开关位于 AC ECU 和执行器之间的供电线路上。当车速小于或者等于 15km/h 时，开关闭合；当车速大于 15km/h 时，开关打开。开关的这种操作与供电线的状态无关（开关本身的供电是独立的）。

——执行器通电后才可被激活。

功能安全要求：

——要求 B1: VS ECU 发出准确的车速信息给 AC ECU。➔ASIL C

——或者：避免不正确的传输车速低于或者等于 15 km/h 的信息。➔ASIL C

——要求 B2：当车速超过 15km/h 时，AC ECU 不能给执行器供电。➔ ASIL X(C)（参见表 4）

——要求 B3: VS ECU 发送准确的车速信息给冗余开关。➔ ASIL C

——要求 B4: 在车速大于 15km/h 时，冗余开关处于打开状态。➔ ASIL Y(C)（参见表 4）。

——要求 B5:执行器只有当 AC ECU 给出供电并且冗余开关闭合时才工作。➔ ASIL C

为了允许ASIL分解，如果必要，开发者需增加一个独立性要求：

——要求 B6: 说明 AC ECU 和冗余开关之间的充分独立性。➔ ASIL C

起初的要求A2被冗余要求B2和B4所替代，后两者都遵从安全目标，因此可以应用ASIL 分解。

表4可能的分解

	要求 B2: ASIL X(C)	要求 B4: ASIL Y(C)
可能1	ASIL C(C) 要求	QM(C) 要求
可能2	ASIL B(C) 要求	ASIL A(C) 要求
可能3	ASIL A(C) 要求	ASIL B(C) 要求

可能4	QM(C) 要求	ASIL C(C) 要求

附录A

（资料性附录）

GB/T XXXXX和微控制器

A.1 总则

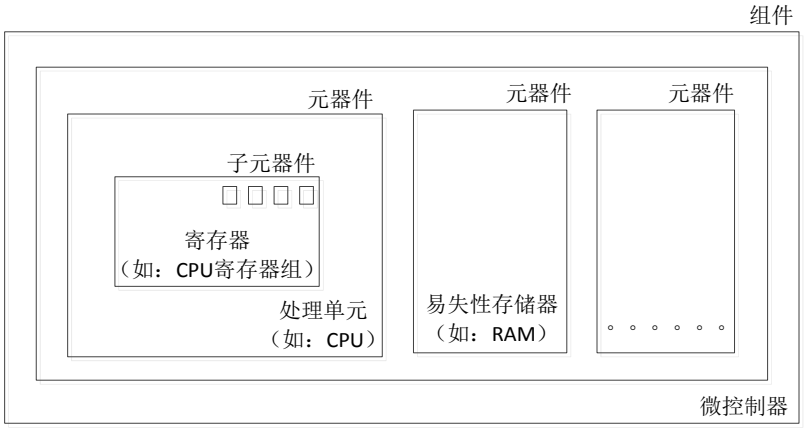
本章的目的是给出在GB/TXXXXX的应用中如何处理微控制器的非详尽示例。

A.2 微控制器，其元器件及子元器件

微控制器（也称MCU或者μ C）是在单一集成电路上由内部中央处理单元（CPU），时钟发生器，定时器，外围设备，输入输出端口和存储器组成的微小计算机。非易失性存储器形式的程序存储器（如Flash或者OTP ROM）和一定容量的随机存取存储器（RAM）也经常被集成在芯片上。

如下图所示，整个微控制器架构可以被视为一个组件，处理单元（如CPU）可视为一个元器件。如已在示例4.2中解释过的和在A.3.3章中更进一步细化的，在特定情况下（如取决于在微控制器或者系统层面所采用的安全机制的类型），每一个元器件可以进一步的分为子元器件（如CPU寄存器组和其内部寄存器）。

下图展示了微控制器的一种逻辑视图，既没有必要转化成微控制器的物理实现，也没有必要表示元器件和子元器件的相关性。



图A.1 微控制器，其元器件及子元器件

GB/TXXXXX-5，附录D，尤其是GB/TXXXXX-5中表D.1，给出了微控制器的元器件和子元器件的清单。未包括在GB/TXXXXX-5中表D.1中的元器件和子元器件可以按表A.1给出的一些示例中所定义的元器件和子元器件进行类似的分类。

表A.1 GB/TXXXXX-5中出现过的微控制器的元器件及子元器件的分类示例

GB/TXXXXX-5:20XX中表D.1的要素	微控制器的示例	
	元器件	子元器件

电源	嵌入式电压调节器（EVR），电源管理单元（PMU）	
时钟	锁相环（PLL），环形振荡器，时钟产生单元（CGU），时钟树	
非易失性存储器	闪存（FLASH），只读存储器（ROM），一次性可编程存储器（OTP）	存储单元阵列，地址解码器，接口电路，测试/冗余逻辑电路，存储器控制器
易失性存储器	随机存取存储器（RAM），缓存	存储单元阵列，地址解码器，接口电路，测试/冗余逻辑电路，存储器控制器
模拟输入/输出和数字输入/输出	通用输入/输出端口（GPIO），脉宽调制器（PWM）	
	模拟-数字转换器（ADC），数字-模拟转换器（DAC）	
处理单元	中央处理单元（CPU）	算术逻辑单元（ALU），中央处理单元（CPU）的数据通道
		寄存器组，中央处理单元（CPU）内部的随机存取存储器（RAM）如小数据缓存
		加载/存储单元和总线接口
		定序器、编码器和执行逻辑，包括标志寄存器和栈控制
		中央处理单元（CPU）的本地存储控制器，包括缓存控制器
	中断控制器	中断控制器的配置寄存器
	通用定时器	
通信	包括总线仲裁的片内通信	总线矩阵/交换机结构，协议，数据宽度和时域转换（例如总线桥）
	使用直接存储器存取（DMA）的片内通信	DMA寻址逻辑，DMA寻址寄存器，DMA缓冲寄存器
	串行外设接口（SPI），串行存储接口（SMI），内部集成电路（I2C）接口，控制器局域网络（CAN）接口，时间触发的控制器局域网络（TTCAN），FlexRay，LIN，单边缘渐近传输（SENT），以太网，分布式系统接口（DSI），外围传感器接口（PSI5）	

注：本表是一个示例，元器件和子元器件列表以及微控制器的划分可以是不同的。

A.3 GB/TXXXXX 中微控制器开发及安全分析概览

A.3.1 总则

作为按照GB/T XXXXX开发的相关项中的一个元器件，微控制器是基于由相关项的顶层安全目标得出的安全要求开发的。硬件架构度量和随机硬件失效概率度量的目标值应分配给相关项：在这种情况下，微控制器只是其中的一个要素。如GB/TXXXXX-5中8.2所述，为便于分布式开发，目标值可以分配给微控制器本身。微控制器的安全性分析是基于GB/TXXXXX-5中7.4.3和GB/TXXXXX-9中第8章（安全分析）定义的要求和建议进行的。

另一方面，在目标相关项尚不存在的情况下，微控制器可以作为脱离环境的安全要素（SEooC，参见本部分的第8章）进行开发。在这种情况下，开发是基于假设的微控制器使用条件（用途假设）进行的，假设的有效性是由微控制器的要求确定的，这些要求是由该微控制器将要应用于的相关项的安全目标导出的。

本章的下述部分所描述的分析和相关示例都假设微控制器是一个脱离环境的安全要素（SEooC），即使微控制器不是一个脱离环境的安全要素（SEooC），所述的方法（如微控制器的失效率计算方法）仍然有效。当这些分析在独立的微控制器上实施时，应进行适当的假设。A.3.9章描述了在系统层面如何匹配和验证这些分析和假设。在独立的微控制器级别，GB/TXXXXX-5、GB/TXXXXX-8和GB/TXXXXX-9的每个要求（例如安全分析、相关失效分析、验证等）仍然有效。

A.3.2 微控制器的定性和定量分析

如GB/T XXXXX-9中8.2所述，在概念和产品开发阶段中适当的抽象层面执行定性和定量的安全分析。就微控制器而言：

a) 定性分析有益于识别失效。一个可能的方法是可以使用来自微控制器框图的信息和来自 GB/T XXXXX-5 中附录 D 的信息进行分析。

注1：GB/T XXXXX-5中附录D可作为声明的有合适理由支持的诊断覆盖率（DC）的起点。

注2：定性分析包括本部分A.3.6（相关失效分析的示例）中所述的相关失效分析。

b) 使用下列组合来执行定量分析：

i) 逻辑块层面的架构；

ii) 来自微控制器的寄存器传输级（RTL）描述信息（以获取功能信息）和门级网表（以获取功能和结构信息）；

iii) 用于评估子功能间未定义的潜在交互的信息（相关失效，见A.3.6章）；

iv) 布局信息-只有在最后阶段才可用；

v) 用于验证有关某些特定的故障模型（如桥接故障）的诊断覆盖率的信息。这只适用于某些情况，如一个元器件和其相应的安全机制之间的比较；

vi) 有证据支持的，以及在审慎考虑了系统层面措施有效性后的专家判断。

注1：相关失效分析是在定性的基础上进行的，因为不存在普遍的和足够可靠的方法来量化此类失效。

注2：在微控制器开发阶段可逐步获取此信息。因此，分析可基于最新信息重做。

示例1：在设计初期启动相关失效的评估。应定义设计措施以避免和揭示相关失效的潜在来源或者探测相关失效对“片上系统”的安全性能的影响。在最终设计阶段进行布局确认。

示例2：前期可测性设计和前期布局门级网表可以用于定量分析的初始阶段；而后期可测性设计和后期布局门级网表可以用于随后重复的定量分析。

c) 由于微控制器的元器件和子元器件可以在一个单一的物理组件中实现，相关失效分析和独立或免于干扰的分析都是微控制器的重要分析。更多详情请参阅 A.3.6 段。

A.3.3 微控制器失效率计算的方法

A.3.3.1 总则

失效率计算的要求和建议定义在GB/T XXXXX-5中，度量计算的要求参见GB/T XXXXX-5附录C。

按照GB/T XXXXXX-5中附录E的示例，微控制器的失效率和度量可用下面的方法计算：

——首先，将微控制器划分为元器件或子元器件。

注1：在相关失效分析中，验证识别出的元器件独立性假设。

注2：必要的详细程度（例如，是否仅细化到元器件层面，还是继续细化到子元器件层面或者基础子元器件层面）可取决于分析的阶段和所使用的安全机制（微控制器内部或系统层面）。

示例1：如果CPU的功能由另一个锁步运行的CPU进行监测，分析不需要考虑每个CPU内部寄存器，但需要考虑锁步比较器更多的细节。另一方面如果CPU的功能是由软件自检监测，详细分析CPU各子元器件是合适的。

示例2：计算的置信度与详细程度成正比：低详细程度在概念阶段分析时较为合适，而更高的详细程度在开发阶段分析时较为合适。

注3：由于现代微控制器（成百上千的元器件和子元器件）的复杂性，为了保证分析的完整性，自动工具有助于支持各分支过程。应小心以确保微控制器级分析触及模块边界。如果寄存器传输级（RTL）是可用的，划分在寄存器传输级（RTL）层次完成。

——其次，用以下两种方法中的一种来计算各元器件或子元器件的失效率：

1）如果整个微控制器芯片（即不包括封装和焊接）的总失效率（FIT值）是已知的，那么元器件和子元器件的失效率可认为等于元器件或子元器件所占的面积（即门、触发器和相关的互接线有关的面积）除以微控制器芯片的总面积再乘以总失效率。

注1：对于带有功率级的混合信号芯片，这种方法适用于各个域内部，因为数字域的总失效率可能与模拟域和功率域不同。

注2：微控制器的详细知识是有用的。

示例：如果一个处理器（CPU）面积占整个微控制器芯片面积的3%，那么它的失效率可以被假定为等于整个微控制器芯片失效率的3%。

2）如果基本失效率，即基础子元器件（如微控制器的门）的失效率是已知的，那么可假设元器件或者子元器件的失效率等于各类基本子元器件的数量乘以其失效率后再求和。

注1：微控制器的详细知识是有益的。

注2：如何得出基本失效率值可参见A3.4的示例。

——通过将故障归类为安全故障、残余故障、可探测的双点故障和潜伏的双点故障来完成评估。

示例：处理器内的调试单元的特定部分是安全相关的（因为处理器本身是安全相关的），但它们自身不能直接导致违背安全目标，或它们的失效不能显著增加违背安全目标的概率。

——最后，确定元器件或子元器件中与残余和潜伏故障相关的失效模式覆盖率。

示例：针对某一特定失效率的失效模式覆盖率的计算，可以通过将子元器件划分为更小的子元器件，并为每个小的子元器件计算预期的安全机制覆盖能力来进行。例如，CPU寄存器组某一失效的失效模式覆盖率可以通过将寄存器组划分为更小的子元器件（例如，每一个小的子元器件都对应特定的寄存

器，如R0，R1，……），并计算它们各自的安全机制的失效模式覆盖率（例如，将相应的低层面失效模式的失效模式覆盖率进行组合）来计算。

注1：安全机制的有效性可能受到相关失效的影响。需考虑足够的措施，如A.3.6节所列。

注2：由于在这一层面的分析中不能考虑车辆驾驶员的故障探测能力，可感知的故障的概念不适用于微控制器层面。有关微控制器层面的信息与应用的结合，更多细节参见A.3.9。

注3：由于现代微控制器的复杂性（数以百万计的门电路），故障注入方法可以协助计算并用于验证安全故障数量，尤其是失效模式覆盖。进一步的详情请参阅A.3.8.2。故障注入并不是唯一的方法，A.3.8.2中描述的其它方法也是可以接受的。

A.3.3.2 如何考虑瞬态故障

如 GB/T XXXXX-5，8.4.7 注 2 所述，当显示瞬态故障是相关的，比如因为所用的技术。要考虑这些瞬态故障。可通过指定并验证一个特定的“单点故障度量”目标值，或通过一个定性理由来处理这类瞬态故障。

当使用定量的方法时，可按照GB/T XXXXX-5中附录E给出的示例计算瞬态故障的失效率和度量，通过以下方法：

——首先，如A.3.3所述，将微控制器划分为元器件或者子元器件。

注：由于RAM存储器中存储要素的数量多、密度高，其瞬态故障的失效率明显高于微控制器的处理逻辑或其它部分的失效率。因此，如GB/T XXXXX-5中8.4.7注1所推荐的，分别为RAM存储器和微控制器的其它部分计算失效率（和度量）是有帮助的。

——其次，使用基本失效率计算每一元器件或者子元器件瞬态故障的失效率。

示例：按照A.3.3中定义的方法，基本失效率可以使用考虑了单粒子翻转、单粒子瞬变以及电路有关部分（例如触发器和门电路的数量）的基本失效率的函数来计算。参见A3.4中有关如何得到基本失效率数值的示例。

——最后，通过将故障分类为安全故障和残余故障来完成评估，即评估与元器件或子元器件失效率相关的安全故障的数量。

注：当安全瞬态故障数量的估算对应用软件有明确的依赖性，且如果该软件在微控制器开发时是不可用的，那么按照50% - 50%来估算安全瞬态故障的数量是可以接受的。当应用软件是可用的或者如果直接依赖于微控制器架构，最好有确定该值的具体分析。

示例：存储安全相关常数（即一个值只写一次，但在每个时钟周期读取，如果错误则违背安全目标）的寄存器的故障永远不是安全的。相反的，例如，如果寄存器每10毫秒写入一次，但只有一次用于安全相关的计算，在它被写入1毫秒后，寄存器中的一个随机瞬态故障会导致90%个安全故障，因为在其余90%个时钟周期内，该寄存器的故障不会导致违背安全目标。

注1：在GB/T XXXXX-5中8.4.7注2所述，瞬态故障可以通过单点故障度量来处理。分析潜伏故障时不需考虑瞬态故障。计算潜伏故障的失效模式覆盖率也不需考虑瞬态故障，因为瞬态故障的根本原因很快就消失了（由瞬态的定义）。此外，假定在绝大多数情况下，瞬态故障的影响会迅速被修复。例如：瞬态故障发生后，在可导致多点失效发生的第二个故障出现前（瞬态故障为第一个故障），通过后续的下电循环，移除触发器或存储单元中由于瞬态故障引发的错误状态。在特殊情况下，这可能无效，需要额外的措施，但此类情况可通过具体问题具体分析来解决。

注2：瞬态故障仅限于受影响的子元器内，如果其中没有逻辑连接，并不会扩散至其它子元器件。

注3：在GB/T XXXXX-5中表D.2到D.14里定义的某些安全机制的覆盖率仅对永久性故障有效。这种重要的区别，可以在相关安全机制的描述中发现，描述中写明了如何考虑瞬态故障的的覆盖率值。

示例：RAM跨步测试（GB/T XXXXX-5中表D.6）覆盖率的典型值被评为高。然而在相关描述中（GB/T XXXXX-5中D.2.5.3）写明了该类型的测试对软错误的探测是无效的。例如，RAM跨步测试对瞬态故障的覆盖率为零。

当使用定性的方法时，应给出基于验证安全机制有效性的理由，安全机制是为了覆盖瞬态

故障而实施（无论是微控制器内部或是系统层面）的。

示例：对于数据路径要素，数据处理的时间冗余（即多次处理相同的信息）已能确保对瞬态故障的高级别保护。

A.3.4 如何得出可用于微控制器的基本失效率

A.3.4.1 总则

如GB/T XXXXX-5中8.4.3中所述，失效率数据可以由业界公认的来源导出。下面举例列出了可以为A.3.3和A.3.2中定义的方法提供基本失效率来源的标准和手册：

——对于永久性故障：使用半导体行业提供的数据或使用标准如IEC / TR 62380 [8]和SN 29500 [6]的数据；

注：对于永久性故障：半导体行业提供的数据可以基于（随机）失效的数量除以等效设备小时而得出。这些数据来自现场或来自加速寿命试验（如在标准JEDEC和AEC中所定义）。加速寿命试验是按比例缩放至某一任务剖面（如温度，开/关循环），并假设失效率（随机失效，成指数分布）为定值。基于抽样统计的置信度水平，这些数字可以被假设为最大的FIT值。

——对于瞬态故障：使用半导体行业提供的从JEDEC标准（如JESD89）和国际半导体技术路线图（ITRS）得出的数据。

注：如果有适当的证据支持，可以通过考虑其它因素（诸如寄存器密度和钥匙开启和关闭之间永久性故障的发生概率）对由标准和手册得到的基本失效率进行调整。

A.3.4.2 根据IEC / TR 62380计算微控制器芯片失效率（FIT）的示例

GB/T XXXXX-5中8.4.3中声明的失效率数据可以由行业公认的来源导出，例如IEC / TR 62380、IEC 61709、MIL-HDBK-217 F-notice 2、RIAC-HDBK-217 Plus。以下是使用IEC / TR 62380 [8]中详细定义的定量分析方法估算硬件失效率（FIT）的示例。在IEC / TR 62380半导体的失效率（FIT）模型中，认为设备的失效率等于微控制器芯片（不包括封装和焊接）失效率、封装失效率和接口的电气过应力引起的设备失效率之和。

注：1 FIT对应设备运行时间每10^9小时一个失效。

A.3.4.2.1 微控制器芯片的失效率计算示例

为了计算基础微控制器芯片的失效率（FIT）（即根据运行条件，对失效率进行修正前），需考虑四个关键要素：

——λ_1，每个晶体管的基本失效率（FIT），取决于工艺技术；

——N，所用晶体管的数量；

——α，工艺成熟度的修正因子；随着工艺技术的成熟，每个晶体管的失效率呈指数降低到渐近水平；

——λ_2，受工艺技术影响的失效率（FIT），与晶体管的数量或寿命无关。

这些因子按IEC / TR 62380:2004中7.3.1章中的公式（“数学模型”）进行结合。

参数的选择可以基于设计中所用的工艺技术和电路类型来完成。在IEC/TR 62380:2004中表16提供的值可用于CMOS逻辑电路、模拟电路和多种类型的存储器（SRAM、DRAM、EEPROM、Flash EEPROM等）。

表A.2展示了A.3.5中所用的定量示例中失效率的计算。对于工艺成熟度的修正因子，假设2008年是制造年份。

表A.2 失效率的计算示例

电路要素	λ_1	N	α	λ_2	基本失效率（FIT）
50k门CPU	3.4×10^{-6}	200000(4晶体管/门)	10	1.7	1.72
16kB SRAM	1.7×10^{-7}	786432（6晶体管/位低功耗SRAM）	10	8.8	8.802
总和					10.52

注1：对于某个给定的电路类型，λ_1 和 λ_2可能有多个有效值。在这种情况下，评估方应确保所选的值与所用特定制造技术的度量最为匹配，并提供适当的理由。

注2：为了简化计算，对整个器件的估算可选用单一λ_1 和 λ_2。

注3：工艺成熟度修正因子的引入考虑了摩尔定律和器件失效率保持不变的事实。如果每个晶体管的失效率保持不变，则根据摩尔定律，总失效率将会增加。但这样的情况并没有被观测到。因此，改变工艺节点时，晶体管的失效率不会保持不变。IEC/TR 62380:2004建议使用制造日期。为了反映工艺技术的变化，也可以选择首次引入这个特定技术节点的年份以来替代晶体管的制造年份。为了不受硅供应商工艺差异的影响，可以使用半导体国际技术路线图（ITRS）中的年份。

注4：为了计算整个器件中微控制器芯片的失效率，使用等效门电路的数量。有效的等效晶体管的数量等于等效门电路数量乘以每个门电路里的晶体管数量的典型值。当计算由于CMOS数字逻辑所导致的微控制器芯片的失效率时，模块（例如CPU、CAN、定时器、FlexRay、SPI）内每个数字逻辑对失效率的贡献被包含在N里。

注5：对于模拟元器件和以模拟工艺技术为主的微控制器，可以使用IEC/TR 62380:2004中表16开始处“MOS : Standard circuits (3)”中“LinearCircuits”的数据，除非微控制器供应商提供了更精确的数据。

注6：为了计算I/O对失效率（FIT）的贡献时，可以按照注4中CMOS数字逻辑的等效晶体管数量考虑，除非微控制器供应商提供了更精确的数据。

微控制器芯片的基本失效率（FIT）一旦产生，将应用基于热效应和运行时间的修正因子。修正因子考虑了：

——微控制器芯片的结温，这基于以下计算：

——微控制器芯片的功耗；

——封装的热阻，基于封装类型，封装引脚数和气流；

定义了1～Y使用阶段的应用剖面，每个剖面由“使用时间占器件总寿命的百分比”和“环境温度”组成。IEC/TR 62380:2004提供了“发动机控制”和“乘客舱”两种可供参考的汽车应用剖面。

——由技术类型决定的活化能和频率以完成阿伦尼乌斯方程。

在本例中，我们假设一个基于CMOS技术的微控制器的功耗是0.5W。微控制器芯片封装在一个144引脚方型扁平式封装（QFP）中，并通过自然对流冷却。该微控制器被暴露在“发动机控制”的温度剖面中。由此而增加的结温ΔT_j是26.27℃。由阿伦尼乌斯方程假定的活化能是0.3 eV。使用IEC/TR 62380:2004中的修正公式，会得到一个值为0.17的修正因子。

当应用修正因子时，可以得到如表A.3所示每个部分的有效失效率。

表A.3 每个部分的有效失效率（FIT）的示例

电路要素	基本失效率（FIT）	温度修正因子	有效失效率（FIT）
50K门CPU	1.72	0.17	0.29

16KB SRAM	8.8	0.17	1.5
总和			1.79

注：为了更准确的估算失效率（FIT），可以考虑使用产品特有的诸如封装的热特性、制造工艺、阿伦尼乌斯方程等数据，以替代IEC/TR 62380:2004中的通用因数值。

A.3.4.2.2 IEC / TR 62380中修正因子的另一种计算方法

即使IEC/TR 62380:2004提供的失效率与当前的可靠性数据更为一致，使用更为保守的数据也是有益的，可通过使用例如SN29500等更早的失效率手册，也可通过微调所用的温度修正因子来实现。

在IEC/TR 62380:2004中7.3.1用来计算温度修正因子δ_T的公式（“数学模型”）使用以下参数：

$(\pi_t)_i$：与集成电路的任务剖面中第i个结温相关的第i个温度因子。

τ_i：任务剖面中集成电路的第i个结温对应的第i个工作时间比例。

τ_{on}：集成电路的总工作时间比例，由$\tau_{on} = \sum_{i=1}^{y} \tau_i$。

τ_{off}：集成电路的存储（或休眠）的时间比例。

$$\tau_{on} + \tau_{off} = 1$$

对于保守的温度修正因子的计算，存储时间τ_{off}可以设置为零，由此会得出温度修正因子δ_T的小修改版本$\delta_{T,保守}$：

$$\delta_{T,保守} = \frac{\sum_{i=1}^{y} (\pi_t)_i \times \tau_i}{\tau_{on}}$$

应用这些公式可得出温度修正因子值为2.91，从而得到表A.4所示的结果。

表A.4 每个部分的有效失效率（FIT）示例

电路要素	基本失效率（FIT）	温度修正因子	有效失效率（FIT）
50K门CPU	1.72	2.91	5.01
16KB SRAM	8.8	2.91	25.61
总和			30.62

A.3.4.2.3 封装失效率的示例

封装失效率由IEC/TR 62380:2004中7.3.1所示的公式（“影响因子π_α的数学表达式”）计算，并使用下列参数：

π_α：与安装基板和封装材料之间热膨胀系数差值相关的影响因子。

$(\pi_n)_i$：在温度变化为ΔT_i的范围内，与封装处年度内温度变化循环次数相关的影响因子。

ΔT_i：任务剖面中第i个温度变化范围。

λ_2：集成电路封装的基本失效率。

在本例中，我们假设一个基于CMOS技术的微控制器的功耗是0.5W。微控制器芯片封装在

一个144引脚方型扁平式封装（QFP），并通过自然对流冷却，由此而增加的结温Δ T_j是26.27℃。该微控制器被暴露在“引擎控制”的温度应用中。

影响因子π_α可用IEC/TR 62380:2004中7.3.1所示的公式（“Mathematicalexpression of the influence factor"）计算，基板和组件各自的线性热膨胀系数分别是α_s和α_c。在本例中，我们假设FR4是安装基板和塑料封装，由该表查得α_s= 16和α_c= 21.5。

在汽车应用中，如果年度内温度变化循环次数≤8760，$(\pi_n)_i$是用IEC/TR 62380:2004中7.3.1的公式计算（"Mathematical expression of the influence factor$(\pi_n)_i$”），n_i是温度变化为ΔT_i的范围内的年度温度循环次数。

为了计算失效率（FIT）中的λ_2，使用外围连接封装的计算公式，如IEC/TR 62380:2004中表17b所示，其中宽度值为20mm，管脚中心距为0.5mm。

使用“发动机控制”的温度剖面，这将导致封装的总失效率为

$\lambda_{封装}$ = 207 FIT

封装失效率在引脚间均匀分配，得到一个引脚的失效率为

$\lambda_{引脚}$ = 1,44 FIT

注：封装失效率的估算是基于设备封装和系统的印刷电路板的结构和热特性的知识。除了使用IEC/TR 62380:2004，还可使用由微控制器供应商和系统制定者对封装失效率（FIT）的联合保守估计。

A.3.4.2.4 电气过应力导致的失效率示例

由电气过应力导致的整个器件的失效率可以用IEC/TR 62380:2004中7.3.1所示的公式计算（“MATHEMATICAL MODEL”）。如果器件具有对外部环境的直接连接，即器件是个接口，π_I等于一；如果器件不是一个接口，即它对外部环境没有直接的关系，π_I等于零。

IEC/TR 62380:2004提供了各种电气环境下不同的λ_{EOS}，但是并没有给出汽车的电气环境，则可以选择“民用航空电子设备（机载计算器）”：

λ_{EOS}= 20 FIT

整个器件由电气过应力导致的失效率是：

如果器件有直接接触外界环境，$\lambda_{过应力}$ = 20FIT；

在其它任何情况下，$\lambda_{过应力}$ = 0 FIT。

预测电气过应力对器件的影响是有意义的。如果没有特别的影响，$\lambda_{过应力}$可以添加到$\lambda_{芯片}$以增加整个器件的微控制器芯片的总失效率。

注：在一些分析标准里，电气过应力被认为是系统失效模式，对于随机失效度量的计算，电气过应力的失效率（FIT）可以降低到零。

A.3.5定量分析示例

下面是一个使用A.3.3中所描述的方法进行定量分析的示例。

注1：在这个示例中使用的数字（例如失效率、安全故障数量和失效模式覆盖率）只是示例。它们随架构不同而不同。

注2：以下示例将微控制器的一部分细分到子元器件的级别。如在A.3.3中讨论过的，必要的详细程度取决于分析的阶段和所用的安全机制。

注3：以下示例使用定量的方法为瞬态故障计算一个专用的“单点故障度量”目标值。如在A.3.3.2中讨论过的，

瞬态故障也可以采用定性的理由来处理。

本例仅考虑微控制器的一小部分，即只有2个元器件：

a) 一个小的CPU，分为五个子元器件：寄存器组、累加器（ALU）、加载存储单元、控制逻辑单元和调试单元。每个子元器件又进一步分为几个子元器件。

b) 一个16K的RAM分为三个子元器件：单元阵列、用于下线检测的地址译码器和逻辑单元、内存备用空间（冗余）的管理。

注1：示例中所示的失效率（FIT）不包括外围设备或其它特性，如封装、搬运（操作、处理）或过应力（过载）。只是给出了作为计算失效率（FIT）的一个可能的方法。出于这个原因，这些值与SN29500中示例所给出的有完整封装的微控制器的失效率（FIT）不能相比。

注2：以下示例的目的是为了避免在系统层面分析中处理每一个最小的微控制器子元器件的要求。在系统层面分析中，组件或元器件层面的细节就足够了。本示例的目的是为了说明可能有必要在独立的微控制器层面进行深入的分析（例如，在子元器件层面），按照所需的精度计算器件和子元器件的失效率和失效模式覆盖率，以供系统工程师使用。也就是说，没有准确并详细的独立的微控制器层面分析，很难有合适的数据进行系统层面分析。

考虑以下四种安全机制：

1）对CPU程序顺序进行逻辑监控的硬件安全机制（SM1）。这种安全机制能够以一定的覆盖率探测到控制逻辑内可能会导致软件顺序错误的故障。但是，这种安全机制难以发现导致数据错误的故障（如错误的算术运算）。

注：本示例中，假定探测到的每个影响中央处理器（CPU）的永久性单比特故障已经发信号给系统（例如通过激活微控制器的一路输出信号）。系统层面的要求是正确使用这个信号（例如，进入安全状态并通知驾驶员）。对于可疑的瞬态故障，中央处理器（CPU）可以尝试通过复位来消除这些故障。如果故障仍然存在，这意味着它是永久性的，因此，可以如前所述，将故障信号发给系统。如果故障消失（即它确实是瞬态的），中央处理器（CPU）可以继续执行任务。

2）用来处理随机硬件失效的基于软件的安全机制SM2。此安全机制在每次钥匙启动时执行，以验证CPU程序顺序的逻辑监控（SM1）不存在潜伏故障。

3）EDC（SM3），可实现RAM单比特位错误纠正和双比特位错误探测。

注：在本示例中，假定每个探测到的永久性单比特位故障（即使能通过EDC纠正）会发信号给软件（例如通过一个中断），并且软件会进行相应的应对。系统层面的要求是正确使用这个事件（例如，进入安全状态并通知驾驶员）。对于可疑的由EDC纠正的瞬态故障，中央处理器（CPU）可以尝试通过在内存中写回正确值来消除这些故障。如果故障仍然存在，这意味着它是永久性的，因此，可以如前所述，将故障信号发给系统。如果故障消失（即它是瞬态的），中央处理器（CPU）可以继续执行任务。为了区分间歇性故障和瞬态故障，一个可行的办法是对纠正次数进行计数。

4）用来处理随机硬件失效的基于软件的安全机制SM4。此安全机制在每次钥匙启动时执行，以验证RAM EDC（SM3）不存在潜伏故障。

为了更好的可视性，表A.5分为三类独立的计算。

表A.5给出了子元器级的失效模式的概览。表A.6展示了如何识别低层级的失效模式以及由此如何计算整体的失效分布，并在后续A.3.9中叙述了方法。

示例：如表A.6所示，触发器x1及其相关的扇入的永久性故障的失效率是0.01FIT。将这些低层级的失效率求和，把累加器（ALU）逻辑的永久性故障的失效率（0.07FIT）作为一个整体进行计算是可行的。用同样的方法，将每个子元器件相关的失效率求和，计算出累加器（ALU）中永久性故障的失效率（FIT）是可行的。

注1：沿失效模式的抽象树向上（即从低层级的失效模式到高层级的），可以组合不同子元器件的失效模式的失效率以计算更高层级的失效模式的失效率，尤其是以更通用的方法定义更高层级的失效模式。

示例：如果一个更高层级的失效模式（如器件层面）被定义为“被CPU处理的错误指令”，失效模式的失效

率可以是子元器件层面上多种失效模式的失效率的组合，如流水线的永久性故障，寄存器组的永久性故障等。因此，如果低级别的失效率是可用的，更高级别的失效率可以用自下而上的方法计算（假定为独立故障）。

注2：表A.5和表A.6的列与7.1.7中描述的故障分类流程图和故障类的基值计算是相关的：

——失效率（FIT）等于λ；

——安全故障数量等于F_{safe}；

——与违背安全目标有关的失效模式覆盖率等于$K_{FMC.\ RF}$；

——残余或单点故障的失效率等于λ_{SPF}或λ_{RF}，取决于失效是单点还是残余。本例中，不考虑单点故障，所以失效率总是等于λ_{RF}；

□ □与潜伏失效有关的失效模式覆盖率等于$K_{FMC.\ RF}$；和

——潜伏多点故障的失效率等于λ_{MPF}。

表A.5 定量分析示例（子元器件层面）

元器件	子元器件		安全相关组件？非安全相关组件？	失效模式	永久失效								瞬态失效				
					失效率 (FIT)	安全故障数量（参见注1）	防止违背安全目标的安全机制	关于违背安全目标的失效模式覆盖率	残余或单点故障失效率（FIT）	防止潜伏故障的安全机制	关于潜伏失效的失效模式覆盖率	潜伏多点故障的失效率（FIT）	失效率 (FIT)	安全故障的数量（参见注1）	防止违背安全目标的安全机制	关于违背安全目标的失效模式覆盖率	残余或单点故障失效率（FIT）
中央处理器（CPU）	寄存器组	寄存器R0	安全相关	永久故障	0.0029	0%	SM1	40%	0.00174	SM1	100%	0.00000					
				瞬态故障									0.03201	0%	SM1	40%	0.01920
		寄存器R1	安全相关	永久故障	0.0029	0%	SM1	40%	0.00174	SM1	100%	0.00000					
				瞬态故障									0.03201	0%	SM1	40%	0.01920
		寄存器R2	安全相关	永久故障	0.0029	0%	SM1	20%	0.00232	SM1	100%	0.00000					
				瞬态故障									0.03201	0%	SM1	10%	0.02880
		寄存器R3	安全相关	永久故障	0.0029	0%	SM1	20%	0.00232	SM1	100%	0.00000					
				瞬态故障									0.03201	0%	SM1	10%	0.02880
	累加器（ALU）	累加器（ALU）	安全相关	永久故障	0.0348	0%	SM1	20%	0.02784	SM1	100%	0.00000					
				瞬态故障									0.00038	20%	SM1	10%	0.00027
		乘法器（MUL）	安全相关	永久故障	0.0290	0%	SM1	20%	0.02320	SM1	100%	0.00000					
				瞬态故障									0.00037	70%	SM1	10%	0.00010
		除法器（DIV）	安全相关	永久故障	0.0232	0%	SM1	20%	0.01856	SM1	100%	0.00000					
				瞬态故障									0.00036	70%	SM1	10%	0.00010
	控制逻辑	流水线	安全相关	永久故障	0.0174	0%	SM1	90%	0.00174	SM1	100%	0.00000					
				瞬态故障									0.00103	20%	SM1	90%	0.00008
		序列发生器	安全相关	永久故障	0.0406	0%	SM1	90%	0.00406	SM1	100%	0.00000					
				瞬态故障									0.00307	50%	SM1	90%	0.00015
		栈控制	安全相关	永久故障	0.0029	0%	SM1	70%	0.00087	SM1	100%	0.00000					
				瞬态故障									0.00033	50%	SM1	40%	0.00010
	加载存储单元	地址生成	安全相关	永久故障	0.0174	0%	SM1	60%	0.00696	SM1	100%	0.00000					
				瞬态故障									0.00103	10%	SM1	60%	0.00037
		加载单元	安全相关	永久故障	0.0145	0%	SM1	50%	0.00725	SM1	100%	0.00000					
				瞬态故障									0.00035	10%	SM1	50%	0.00016
		存储单元	安全相关	永久故障	0.0145	0%	SM1	50%	0.00725	SM1	100%	0.00000					
				瞬态故障									0.00035	10%	SM1	50%	0.00016
	调试	调试内部逻辑	安全相关	永久故障	0.0058	20%	无	0%	0.00464	无							
				瞬态故障									0.00017	20%	无	0%	0.00014
		调试接口	非安全相关	永久故障	0.0783												
				瞬态故障									0.00164				
					Σ				0.11049			0.00000					0.09764
					失效率总计				0.29000			失效率总计	0.13708				
					安全相关总计				0.21170			安全相关总计	0.13545				
					非安全相关总计				0.07830			非安全相关总计	0.00164				
									单点故障 47.80%								单点故障 27.91%
												潜伏故障 100.00%					
易失存储器	RAM (16KB)	RAM数据位	安全相关	永久故障	1.5000	0%	SM3	96.90%	0.04688	SM3	100%	0.00000					
				瞬态故障									131.072	0%	SM3	99.69%	0.40894
		地址译码器	安全相关	永久故障	0.0087	0%	无	0%	0.00870								
				瞬态故障									0.00034	0%	无	0%	0.00034
		测试/冗余	安全相关	永久故障	0.0058	50%	无	0%	0.00290								
				瞬态故障									0.00033	90%	无	0%	0.00003
					Σ				0.05848			0.00000					0.40931
					失效率总计				1.51450			失效率总计	131.07				
					安全相关总计				1.51450			安全相关总计	131.07				
					非安全相关总计				0.00000			非安全相关总计	0.00000				
									单点故障 96.10%								单点故障 99.69%
												潜伏故障 100.00%					
安全机制	SM1	探测逻辑	安全相关	永久故障	0.0029	0%				SM2	90%	0.00029					
				瞬态故障									0.00011				
		警报生成	安全相关	永久故障	0.0029	50%				SM2	90%	0.00015					
				瞬态故障									5.5E-05				
	SM3	EDC编码器	安全相关	永久故障	0.0029	0%	SM3	90%	0.00029	SM4	90%	0.00026					
				瞬态故障									0.00033	0%	无	0%	0.00033
		EDC解码器	安全相关	永久故障	0.0029	0%	SM3	90%	0.00029	SM4	90%	0.00026					
				瞬态故障									0.00033	0%	无	0%	0.00033
		警报生成	安全相关	永久故障	0.0029	50%				SM4	90%	0.00015					
				瞬态故障													
		RAM EDC位	安全相关	永久故障	0.32813	0%	SM3	96.90%	0.01025	SM4	90%	0.03179					
				瞬态故障									28.672	0%	SM3	99.69%	0.08946
					Σ				0.00058			0.03289					0.09011
					失效率总计				0.34263			失效率总计	28.6728				
					安全相关总计				0.34263			安全相关总计	28.6728				
					非安全相关总计				0.00000			非安全相关总计	0.00000				
									单点故障 99.80%								单点故障 99.69%
												潜伏故障 90.40%					

注1：安全故障数量是失效模式的一部分，这些失效模式既没有当安全机制不存在时违背安全目标的可能也没有与另一个子元器件的独立失效组合的可能。

注2：失效模式覆盖率的计算需要详细分析SM1覆盖每一个子元器件的能力。在本示例中，R0和R1是由编译器选择以传递函数参数的寄存器，所以它们导致程序顺序错误的概率稍高，这些错误是可被SM1探测的。本示例的目的是提供证据表明，通过详细的分析来识别子元器件覆盖率的差异是可能的。

注3：例如，如表A.6所示，EDC（SM3）的失效模式覆盖率的计算，需结合EDC探测单、双比特位错误具有高概率和探测多比特位错误的具有低概率（可能小于90%）进行分析。

注4：某些子元器件可以被多个安全机制覆盖：在这样的情况下，将通过详细分析确定的每一个失效模式的覆盖率进行组合，得到失效模式覆盖率。

注5：示例表明，如果没有EDC对多比特位错误的适当的覆盖率，也没有RAM地址解码器的适当的覆盖率，很难达到一个高的单点故障度量。

注6：示例表明，一些安全机制可能会直接导致违背安全目标，因此，在计算残余故障时需考虑它们。本示例中，EDC（SM3）的故障可能在内存没有故障的情况下破坏任务数据。

注7：示例表明，在微控制器中，可能共存有潜在非安全相关的子元器件，但不可能将它们从安全相关的子元器件（调试的内部逻辑）内明确的分离或区分。然而，其它元器件（调试接口）可以很容易地隔离和禁用，某种程度上可以被认为是非安全相关的并且没有风险。

表A.6 定量分析示例（低层级失效层面）

元器件	子元器件	基本子元器件	安全相关组件? 非安全相关组件?	失效模式	永久失效								瞬态失效				
					失效率 (FIT)	安全故障数量	防止违背安全目标的安全机制	关于违背安全目标的失效模式覆盖率	残余或单点故障失效率（FIT）	防止潜伏故障的安全机制	关于潜伏失效的失效模式覆盖率	潜伏多点故障的失效率（FIT）	失效率（FIT）	安全故障的数量	防止违背安全目标的安全机制	关于违背安全目标的失效模式覆盖率	残余或单点故障失效率（FIT）
中央处理器（CPU）	累加器（ALU）	累加器（ALU）	安全相关	触发器X1及其扇入的永久故障	0.0100	0%	SM1	20%	0.00800	SM1	100%	0.00000					
				触发器X1及其扇入的单粒子翻转(SEU)和单粒子瞬态(SET)									0.0001	0%	SM1	10%	0.00009
				触发器X2及其扇入的永久故障	0.0150	0%	SM1	20%	0.01200	SM1	100%	0.00000					
				触发器X2及其扇入的单粒子翻转(SEU)和单粒子瞬态(SET)									0.0001	70%	none	0%	0.00003
				等....	..	..	..	..	..	..	..	..	..	..	..	..	..
				Σ	0.0348				0.02784			0.00000	0.00038				0.00027
易失存储器	RAM (16KB)	RAM数据位	安全相关	相同的编码字中≤2位错误的永久故障	1.3500	0%	SM3	100.0%	0.00000	SM3	100%	0.00000					
				相同的编码字中≤2单粒子翻转(SEU)									129.76128	0%	SM3	100.00%	0.00000
				相同的编码字中>2位错误的永久故障	0.1500	0%	SM3	68.8%	0.04688	SM3	100%	0.00000					
				相同的编码字中>2单粒子翻转(SEU)									1.31072	0%	SM3	68.8%	0.40894
				Σ	1.5000				0.04688			0.00000	131.0720				0.40894

注1：在这种详细程度，才有可能发现某些低层级的失效模式（如触发器X2和其扇入的单粒子翻转和单粒子瞬态故障）是安全的（例如，因为ALU架构几乎不使用那一位）。

注2：对于由单点永久性故障引起的多于两个比特位错误的内存失效率的计算，需要考虑如内存布局信息、地址解码器的结构等内容。

注3：EDC（SM3）对多于两个比特位错误的覆盖率的计算，需要考虑每个编码字的比特位数（本示例中为32）和编码比特位数（本示例中为7）进行详细分析。根据这些参数，覆盖率可以更高。

A.3.6 相关失效分析示例

GB/T XXXXX-9中定义了有关相关失效的识别、评估和解决的一般要求和建议。

相关失效的分析可分为以下步骤：

1）识别相关失效影响到的元器件。

注1：微控制器的安全概念中声称是相互独立的元器件结构，可以很容易地受相关失效的影响。

注2：识别可以通过演绎的安全分析来支持：在双点和多点失效分析中，假定为独立的事件提供了关于元器件易受相关失效影响的有用信息。

2）识别潜在的相关失效源。

要考虑本节列出的主题和其它可预见的物理和逻辑上的相关失效源（共享的逻辑元器件和信号），包括由于不同ASIL等级的功能共存而导致的影响。

3）识别出造成相关失效的元器件之间的耦合机制。

4）定性的列出并评估预防相关失效的措施。

5）定性的列出并评估所采取的设计措施，该设计措施是为了控制在步骤1中识别的每个元器件结构的剩余相关失效所造成的影响。

注：如在GB/T XXXXX-9中7.4.2所注明的，相关失效的分析是基于定性分析，因为对于要量化这样的失效，

没有通用且足够可靠的方法存在。

如在GB/T XXXXX-9中7.4.4注1所述，相关失效的评估可以通过适当的检查清单来支持，即基于现场经验的检查清单。这些清单的目的是为分析师提供关于根本原因和耦合因子的有代表性示例，例如：相同的设计、相同的过程、相同的组件、相同的接口和相同的近似程度。

如GB/T XXXXX-9中7.4.4所述，表A.7列出了相关失效评估要考虑的内容。该表还给出了可以导致相关失效的引发源和耦合机制的非详尽示例，以及避免或探测相关失效的措施的非详尽示例。

注：所列出的措施只是一些可能的选项。避免或探测相关失效的其它措施也是可能的，例如基于系统层面的安全机制。

表A.7 相关失效的评估、潜在的引发源和相关措施的内容

GB/T XXXXX-9中7.4.2.3的内容	潜在引发源和耦合机制的示例	措施示例
硬件失效	能够对元器件和其安全机制都产生违反安全目标的影响的物理缺陷	可通过诸如物理隔离、多样性、生产测试等措施来处理
开发故障	开发阶段引入的有能力造成相关失效的故障，例如：串扰，功能的错误实现，规范错误，微控制器的错误配置等	可通过诸如开发流程定义、多样性、设计规则、配置保护机制等措施来处理
制造故障	制造过程中引入的能够造成相关失效的故障，例如：掩模失准故障	可通过微控制器的详尽的生产测试来处理
安装故障	安装时引入能够造成相关失效的故障，例如：微控制器的印刷电路板连接、相邻元器件的接口等	可通过诸如对ECU的生产测试，安装手册等来处理
维修故障	维修时引入的能够造成相关失效的故障，例如内存的空余行/列内的故障	可通过生产测试，维修手册等来处理
环境因素	典型的环境因素有温度、电磁干扰、湿度、机械应力等。	可通过诸如鉴定测试、压力测试、专用传感器，多样性等措施来处理
共用的内部和外部资源的失效	对微控制器来说，典型的资源有时钟，复位和电源，包括电源分配	可通过诸如时钟监控，内部或外部电源监控，多样分配等措施来处理
特定情况下的应力，如损耗和老化	老化和损耗机制有，例如电子迁移等	可通过设计规则、鉴定测试、多样性、启动测试等来处理

在本节中不包括在微控制器内有潜在能力影响多个元器件行为或者安全机制的共享资源的逻辑失效。它们是标准的定性和定量分析的一部分。

示例：这种类型的典型示例，如DMA控制器、中断控制器、测试逻辑、调试逻辑。

A.3.7 探测或避免微控制器设计中系统失效的技术或措施的示例

GB/T XXXXX-5中7.4.1和GB/T XXXXX-5中7.4.2分别定义了硬件架构设计和详细设计相关的一般要求和建议。此外，GB/T XXXXX-5中7.4.4给出了硬件验证的相关要求。

微控制器的开发是基于标准化的开发流程。下面的两种方法是关于如何提供证据表明在微控制器的开发中采取了足够的措施来避免系统性失效的示例：

a) 使用如表 A.8 中的检查列表；及

b) 通过与目标器件基于同一流程开发的相似产品的现场数据给出依据。

表A.8 在微控制器开发中为符合GB/T XXXXX-5的技术或措施的示例

GB/T XXXXX-5的要求	设计阶段	技术/措施	目的
7.4.1.6 模块化设计特性	设计入口	结构化描述和模块化	以易读的形式描述电路功能，如电路功能可以基于描述而不用进行仿真就能被直观的理解。
7.4.1.6 模块化设计特性		设计描述使用硬件描述语言（HDL）	硬件描述语言里的高级别功能描述，例如VHDL或者Verilog
7.4.4硬件设计的验证		硬件描述语言（HDL）仿真	通过仿真验证VHDL或者Verilog中描述的电路的功能
7.4.4硬件设计的验证		模块级的功能测试（使用例如HDL测试的台架）	自下而上的功能验证
7.4.4硬件设计的验证		顶层的功能测试	微控制器的验证（整个电路）
7.4.2.4鲁棒设计原则		限制使用异步构造	避免整合（综合）过程中典型的时间异常，避免仿真和整合（综合）过程中由不足的建模或可测性设计引起的歧义。这并不排除对特定类型的电路，例如，对复位逻辑或者极低功耗的微控制器，异步逻辑可能是有用的，在这种情况下，目的是提出额外的关注以处理和验证这些电路。
7.4.2.4鲁棒设计原则		主要输入和亚稳性控制的同步	避免因违背配置（时间）和保持时间而引起的电路的不确定表现。
7.4.4硬件设计的验证		以功能和结构覆盖率为导向的验证（使用百分比形式的验证目标覆盖率）	在功能测试时，定量评估所用的验证场景。定义并说明覆盖率的目标等级。
7.4.2.4鲁棒设计原则		编码指导规则的遵守	严格遵守编码风格从而生成句法和语义正确的电路代码。
7.4.4硬件设计的验证		使用代码检查器	由代码检查工具自动验证编码规则（“编码风格”）
7.4.4硬件设计的验证		仿真结果的文档化	对成功仿真所需的全部数据进行文档化，以验证指定的电路功能。
7.4.4硬件设计的验证	综合	为了检查时间限制的门级网表的仿真或者传导延迟的静态分析（STA：静态时间分析）	在综合过程中对达到的时间限制的独立验证。
7.4.4硬件设计的验证		门级网表和参考模型的对比（正式的等价性检查）	综合门级网表的功能等价性检查
7.4.1.6 模块化设计特性		对综合限制、结果和工具的文档化	对最优综合所必须的每个已定义的限制进行文档化，以生成最终门级网表。
7.4.1.6 模块化设计特性		基于脚本的程序	综合循环里的自动化和结果的再现性。

7.4.2.4鲁棒设计原则		对使用少于3年的流程技术的充足的时间余量	即使在流程和参数有大的起伏的情况下，确保所实施的电路功能的鲁棒性。
7.4.1.6 模块化设计特性（可测性）	**测试介入和测试模式生成**	可测性设计（取决于测试覆盖率的百分比）	对于产品测试或者在线测试，为了达到高的测试覆盖率，应避免不可测的或者不易测的结构
7.4.1.6 模块化设计特性（可测性）		自动测试模式生成（ATPG）的测试覆盖率的证据，基于所达到的测试覆盖率的百分比。	在产品测试过程中，对可被综合的测试模式（扫描路径，BIST）所预期的测试覆盖率进行确定。 对覆盖率目标水平和故障模型进行定义和展示。
7.4.4硬件设计的验证		对测试介入后的门级网表进行仿真以检查时间限制，或对传导延迟进行静态分析（STA）	在测试介入时，对所达到的时间限制进行独立验证。
7.4.4硬件设计的验证		对测试介入后的门级网表与参考模型进行对比（形式等价性检查）	对测试介入后的门级网表进行功能等价性检查。
7.4.4硬件设计的验证	**布置、布线和布局生成**	对布局后的门级网表进行仿真以检查时间限制，或对传导延迟进行静态分析（STA）	在后端开发过程中，对所达到的时间限制进行独立验证。
7.4.4硬件设计的验证		电源网络的分析	对电源网络的鲁棒性和相关安全机制的有效性进行展示。例如，电阻压降测试。
7.4.4硬件设计的验证		对布局后的门级网表和参考模型进行对比（形式等价性检查）	对完成后端开发的门级网表进行功能等价性检查。
7.4.4硬件设计的验证		设计规则检查（DRC）	对过程设计规则进行验证。
7.4.4硬件设计的验证		布局与原理图对照检查（LVS）	对布局进行独立验证。
7.4.5 生产、运行、维护和报废 9.4.2.4专用措施	**芯片生产中的安全相关特殊特性**	对生产测试可达到的测试覆盖率进行确定	评估有关微控制器的安全相关方面在生产测试中的测试覆盖率
7.4.5 生产、运行、维护和报废 9.4.2.4专用措施		对探测和清除早期失效的措施进行确定	确保所制造芯片的鲁棒性。在多数（但非所有）过程中，栅氧化层的完整性（GOI）是主要的早期失效机制。栅氧化层的完整性（GOI）早期失效有许多有效的筛选方法：高温/高压运行（老化）、大电流运行、电压应力等。但是，如果栅氧化层的完整性（GOI）不是过程中早期失效的主要原因，同样的方法是无效的。
7.4.5 生产、运行、维护和报废 10硬件集成和测试	**硬件组件的鉴定**	对鉴定测试进行定义和执行，如欠压测试、高温运行寿命测试（HTOL）和功能测试用例	对于集成了欠压探测的微控制器，测试微控制器功能以验证当欠压探测监测到任何供电电压达到正确运行所定义的下边界值时，微控制器的输出会被设置为一个确定的状态（例如在复位状态时停止微控制器的运行）或者用其它方式发出欠压条件信号（例如拉高安全状态信号）。

			对于没有集成欠压探测的微控制器，测试微控制器功能以验证当供电电压从正常值下降到0时，微控制器的输出会被设置为一个确定的状态（例如在复位状态时停止微控制器的运行），否则应定义使用的假设，并且应考虑外部措施。

此外，可以考虑以下的一般准则：

c) 对每项设计活动、功能仿真所用的测试安排和工具及仿真结果进行文档化；

d) 对每项活动及其结果进行验证，例如仿真、等价性检查、时序分析或检查技术约束；

e) 对设计实施过程（基于脚本的、自动化的工作和设计实施流）的可重复措施和自动化措施的使用；以及

注：这意味着有能力冻结工具版本，以具备今后符合法律要求的可重复性。

f) 对第三方软核和硬核，使用经过确认的宏块，并且如果可行的话，遵守每一个约束和由宏核心供应商定义的程序。

A.3.8 微控制器硬件设计验证

A.3.8.1总则

如GB/T XXXXX-5中7.4.4.1所见，按照GB/T XXXXX-8第9章（随机硬件失效导致违背安全目标的评价）验证硬件设计对于硬件安全要求的符合性和完整性。

对于验证，故障注入只是可用的方法之一，也可用其它方法。

示例1：当最新技术水平的解决方案以更高层面的通信要素（例如在IEC 61784中定义的CAN通信的软件层）协议存在时，使用专家判断或者以前证明的结果。

验证方法的选择和验证的深度可基于分析的阶段和使用的安全机制（微控制器内部的或者系统层面的）。

示例2：以下是段落A.3.3中示例1的推理，在完全的硬件冗余情况下（例如使用双核锁步方案中，在每个时钟周期内，两个相同CPU的每个输出由硬件进行对比），对失效模式的覆盖率的验证不需要考虑每个CPU的内部寄存器。相反，对CPU接口和锁步比较器可能需要更详细的验证。

A.3.8.2使用故障注入仿真的验证

如在GB/TXXXXX-5:2011中表3所提到的，在开发阶段，故障注入仿真是验证安全机制对硬件安全要求实施的完整性和正确性的一种有效方法。

以下情况尤其适用：对微控制器的硬件层进行单粒子翻转的故障注入测试是不切实际的或者对特定的故障模型甚至是不可能的。因此，使用设计模式的故障注入（例如：在门级网表进行故障注入）有助于完成此验证步骤。

注1：故障注入可用于永久性（例如：卡滞故障）和瞬态（例如：单粒子翻转）故障。

注2：GB/T XXXXX-5中表D.1表明当没有其它合适的证据时，需考虑直流故障模型（除了卡滞在0和1），能声明对特定要素具有高诊断覆盖率。也写明了并非特意要求对这些故障模型进行穷尽分析，并且如果正确的被使用，由卡滞仿真得到的方法（如N-探测测试，见参考文献[3]-[5]）也是已知验证直流故障模型的有效方法。

示例1：简化直流故障验证的合适方法可以提供证据，证明卡滞在开路/桥接故障的故障分布对于整个直流故障模型总数来说是一个非常有限的部分，例如，远远低于卡滞在0 / 1的故障总数。

示例2：由于不需要穷尽性，直流故障模型分析可以应用到一个微控制器子元器件的子集，该子集的选择取决于直流故障模型对它们的可能影响（例如比较器）或取决于统计基础。

示例3：对于N-探测测试，“正确的被使用”意味着对于相同的故障，模式集（即模式的丰富性）可以确保N个不同的探测。N的范围可以从5到10。

示例4：总体而言，硬件安全机制可以更有效地探测各种直流故障，并且更容易使用诸如N-探测方法对这些机制进行验证。另一方面，在软件安全机制处理随机硬件失效的情况下，由于运行时执行后续测试的环境可能变化，N-探测技术很难在模式的丰富性上获得高的置信度。在这种情况下，可以使用替代的解决方案（如[7]）。

注3：可基于布局分析将故障注入用于向特定位置注入桥接故障；或将故障注入用于验证相关故障的影响，如注入时钟和复位故障。

那些使用设计模式的故障注入可成功地用于协助验证安全故障和计算安全故障的数量及失效模式覆盖率，如在段落A.3.3和A.3.3.2中所见。

示例：注入故障，并在恰当定义的观测点确定故障是否导致可测量的影响。此外，它可以用来协助计算和验证失效模式覆盖率的值，如注入能导致可测量的影响的故障，并确定安全机制是否在故障容错时间间隔内探测到这些故障。

注：用故障注入计算和验证的置信度与用于激励被测电路的测试台架的质量和完整性、注入故障的数量以及电路表示法的详细程度成正比。

示例：门级网表适用于永久性故障的故障注入，如卡滞故障。FPGA类型的方法可有助于将测试执行的速度最大化。对于卡滞故障，只要给出了与门级的关系，“寄存器传输级”也是一个可接受的方法。

A.3.9 如何在系统层面调整和验证微控制器的独立分析

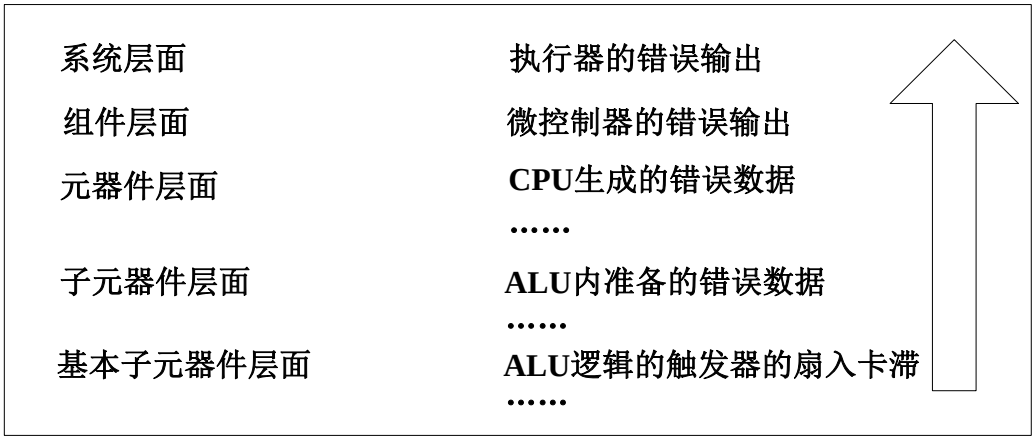
系统层面微控制器的独立分析的调整和验证可通过以下完成：

a) 将微控制器的详细失效模式转换为在系统层面分析时所需的高级别失效模式；

注1：可通过自下而上的过程完成（如下图所示）：使用A.3.2，A.3.3和A.3.5中所描述的方法，识别详细的微控制器失效模式并将其结合到组件层面是可行的。

注2：从抽象的详细程度开始，使微控制器的定量和精确的失效分布成为可能，否则需要基于定性分布假设。

注3：如在A.3.3中所讨论的，必要的详细程度可取决于分析的阶段和所用的安全机制。



图A.2 自下而上的方法来导出系统层面失效模式的示例

b) 在元器件层面或者子元器件层面计算出的失效模式覆盖率可以通过应用层面的措施来提高；

示例：在独立的微控制器级，ADC外设的失效模式覆盖率被认为是零，因为微控制器内部并没有实施安全机制来覆盖这些故障。然而，在应用层，ADC是包含在一个闭环内的，其故障是通过基于软件的一致性检查来探测的。在这种情况下，由于存在应用层的安全机制，该子元器件的失效模式覆盖率可以提高。

c) 在元器件层面或者子元器件层面计算出的失效模式覆盖率是在某些特定的假设(“使用

的假设”）下计算出来的。

注：这种情况下，在应用层面对假设进行验证，如果无效，可以做其它假设并基于这些新的假设重新计算失效模式覆盖率。

示例：在独立的微控制器级，内存的永久性潜伏故障被认为是已探测到的，因为EDC会将每个单位错误纠正标记发给CPU。该假设是基于已经实施了软件驱动来处理这个事件。然而，由于性能原因，这个软件驱动并没实施，因此，假设不再正确。另一种措施是对微控制器进行编程，以直接发送错误纠正标记到外部。可以重新计算内存中的潜伏故障覆盖率。

A.3.10 SEooC微控制器的安全文件示例

如在8.2.3.6所示，对按SEooC开发的微控制器，工作成果的必要信息将被提供给系统集成商，包括假定的要求文档、在SEooC之外并与设计相关的假设和适用的工作成果。

在此基础上，SEooC微控制器的安全文件可以包括按开发接口协议定义的下列文件或子集：

——微控制器相关的安全档案，参见GB/T XXXXX-2，6.5.3；

——微控制器的安全计划，参见GB/T XXXXX-2，6.5.1和GB/T XXXXX-5，5.5；

——GB/T XXXXX-8中适用的其它计划，如配置管理计划、变更管理计划、影响分析和变更需求计划、验证计划、文档管理计划和软件工具的鉴定计划；

——安全计划中适用步骤得到执行的相关证据，如GB/T XXXXX-2所见；

——GB/T XXXXX-5中所见的硬件规范，如硬件安全需求规范、软硬件接口（HSI）规范和硬件设计规范；

——验证计划及其它计划中的适用步骤得到执行的相关报告，如GB/T XXXXX-5和GB/T XXXXX-8所见，例如，硬件安全要求的验证报告、硬件设计验证报告以及硬件集成和验证报告；

——安全分析相关的报告，如GB/T XXXXX-5，GB/T XXXXX-8和GB/T XXXXX-9所见，例如硬件安全分析报告、微控制器架构对处理随机硬件失效的有效性评审报告、随机硬件失效导致违背安全目标的评估评审报告和相关失效分析结果。

注：开发接口协议中应定义需具备哪些文件及以何种详细程度提供给微控制器的客户。

此外，以下信息也是值得收集的：

——关于为微控制器裁剪了的GB/T XXXXX的生命周期的描述；适用的工作成果的清单（GB/T XXXXX生命周期中的哪些工作成果是适用于微控制器的描述）；

——关于微控制器安全架构的描述，包括对微控制器功能的描述和对安全机制的描述；

——关于符合微控制器预期用途的使用假设（AoU）的描述，包括：微控制器安全状态的假设；故障容错时间间隔的假设和多点故障探测间隔的假设；对微控制器应用场景的假设，包括其外部接口；

——关于微控制器配置和在探测到失效后对其进行控制的相关硬件/软件（软硬件）响应过程的描述；

——关于对系统集成商有用的微控制器层面的安全分析结果的描述，如分析中所考虑的故障模型、失效模式及失效率的描述；硬件架构度量（单点故障和潜伏故障度量）；随机硬件失效度量（PMHF）；违背安全目标的每个原因的评估（GB/T XXXXX-5:2011中9.4.3）；相关失效的引发源的描述；为避免或探测相关失效的假定或已实施的措施的描述；安全分析所基于的使用假设（AoU）的描述（例如，基于软件的处理随机硬件失效的安全机制等）；

——关于功能安全评估过程的描述；确认措施的清单和独立等级的描述；微控制器中避免系统性失效的过程总结。

注：该文件可合并到SEooC微控制器的一份名为“安全手册”或“安全应用笔记”的文件中。

附录B

（资料性附录）

故障树的架构和应用

B.1 总则

分析相关项和要素的故障和失效时，最常用的两种技术是FTA和FMEA。FMEA是一种从系统各元器件的失效原因到它们的失效对系统的影响的归纳方法（自下而上，见图B.1）。FTA是一种从非预期的系统行为到导致该行为的可能原因的演绎方法（自上而下，见图B.2）。

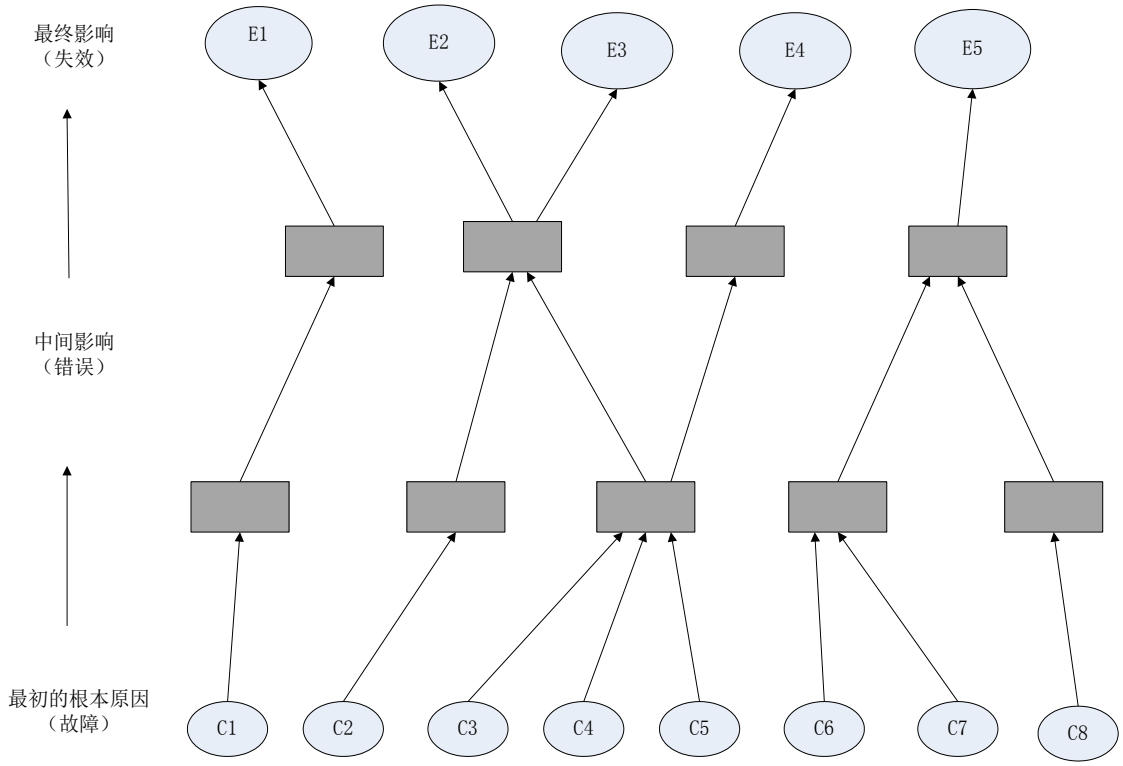
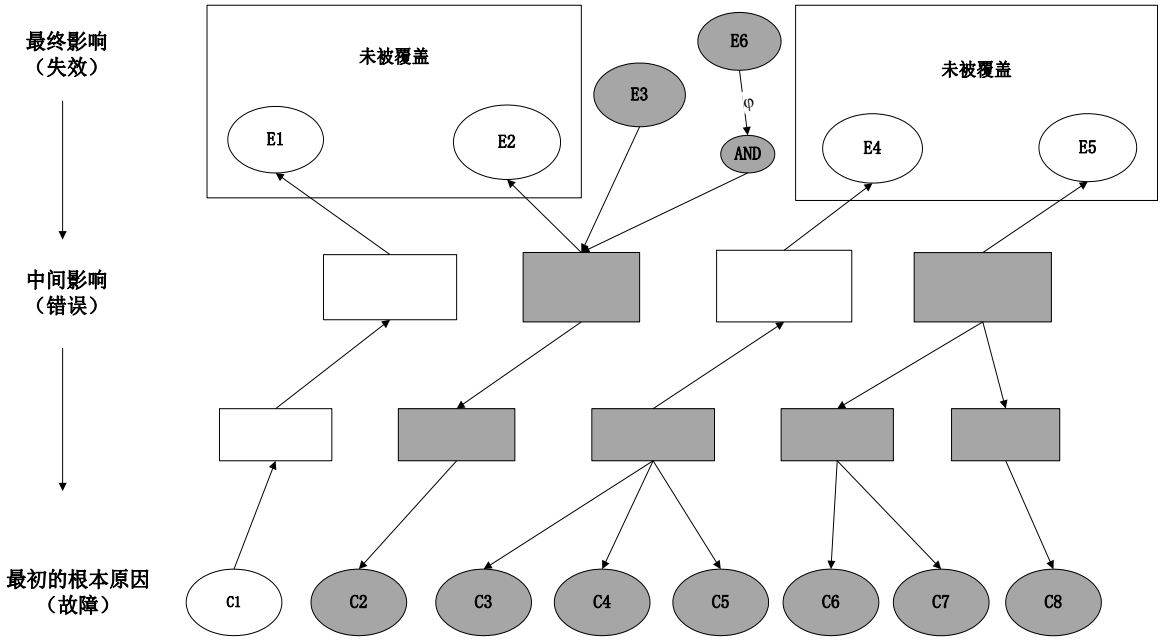


图 B.1 FMEA 图解，自下而上的方法



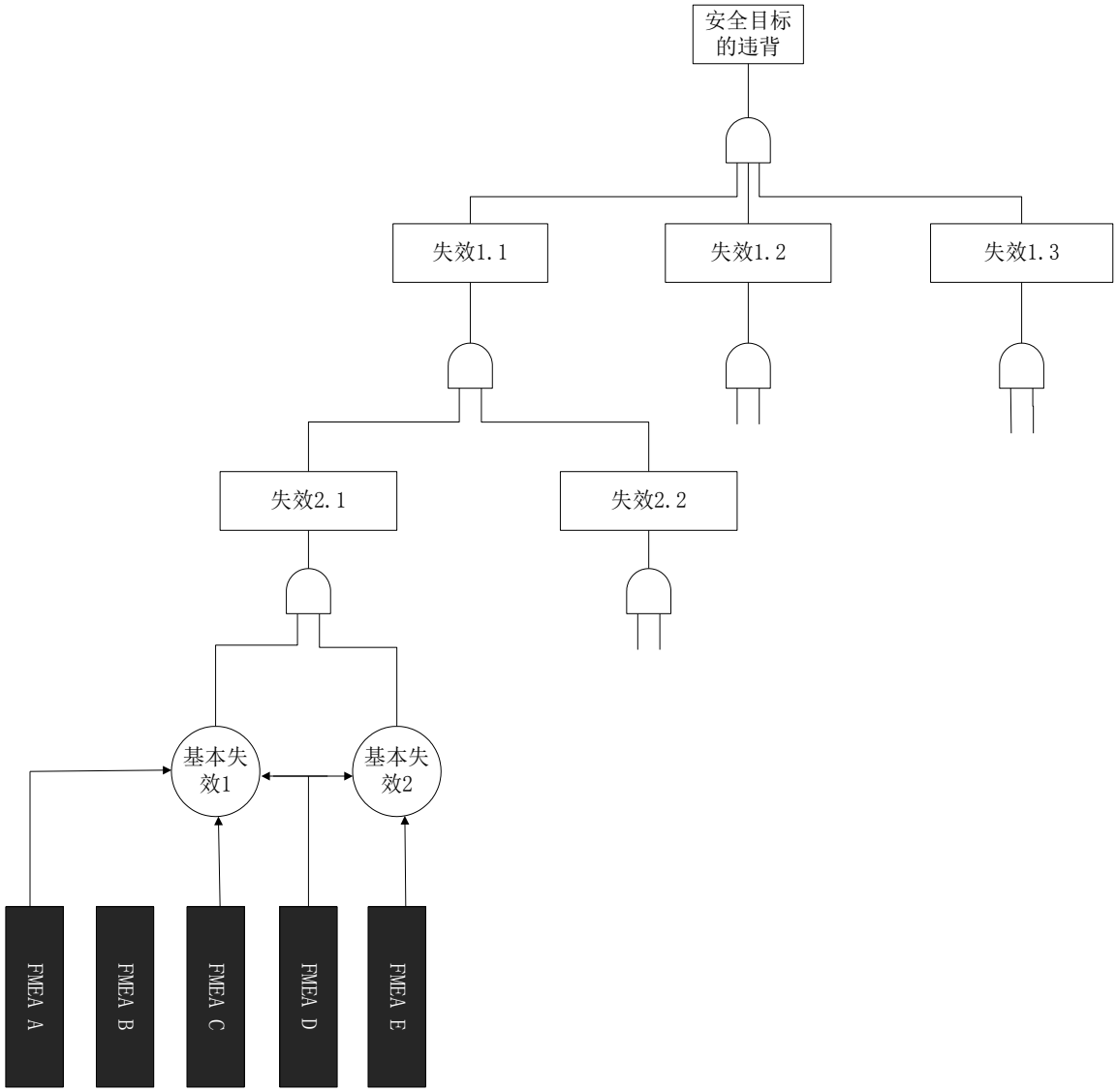
图B.2 FTA图解，自上而下的方法

这些方法是互补的，如GB/T XXXXX-5，7.4.3.1表2注所声明的：“分析的详细程度和设计的详细程度是相称的，在某些情况下，两种方法都可在不同的细节层面上执行”。图B.1和B.2里的椭圆形“Cx”既可表硬件也可代表软件组件。一种典型的分析方法是使用FTA把危害分析到组件层面。然后使用FMEA自下而上的分析组件的失效模式，以确定其失效模式及消除故障树底层故障的安全机制，（宜）避免FTA模型与FMEA的重叠部分所导致的重复工作是所期望的。由此更倾向于将串行系统元器件的FMEA结果做为底事件失效率提供给故障树模型。

注：如GB/T XXXXX-9, 7.4.2所述，因不存在统一的且充分可信的方法来量化相关失效，对相关失效诱因的预估是定性的。因此，本章中的定量方法仅与可量化的相关失效有关，如图B.9，R0瞬态和永久性故障分支中SM1的永久性故障的相同模式诱因。

B.2 FTA 与 FMEA 的结合

系统是由许多零部件和子零部件组成的。可将FTA和FMEA相结合，以提供具有自上而下和自下而上方法合理平衡的安全分析。如图3展示了将FTA和FMEA相结合的可能方法。在此图中，底事件是由在更低的抽象层面(例如：子元器件、元器件或组件层面)上完成的不同FMEA（在本示例中以FMEA A-E标示）中得出的。在本示例中，FMEA B对底事件1和底事件2没有影响，但是FMEA D对它们两个有影响。



图B.3 FTA和FMEA相结合的图解

B.3 故障树示例

B.3.1 总则

以附录A中的微处理器为例构建故障树，这个例子不是关于FMEA和FTA的结合，而是关于如何构建故障树。

通过把表A.5的每一行转换成（故障）树的每一个分枝来构建故障树。附录中图B.6-B.21涵盖了完整的故障树，该故障树示例用来说明两种评估违背安全目标的残余风险是否足够低的方法：基于其违背安全目标的概率，对故障树的分支进行评估。因此不需要计算最高层面违背安全目标的概率。

故障树不用于确定诊断覆盖率、单点和潜伏故障的度量。一旦通过例如FMEA确定了诊断覆盖率，就可将其输入给故障树，由此可计算整个系统生命周期的失效率。

图B.4展示了FTA中单点、残余和双点故障之间的关系的一般示例。

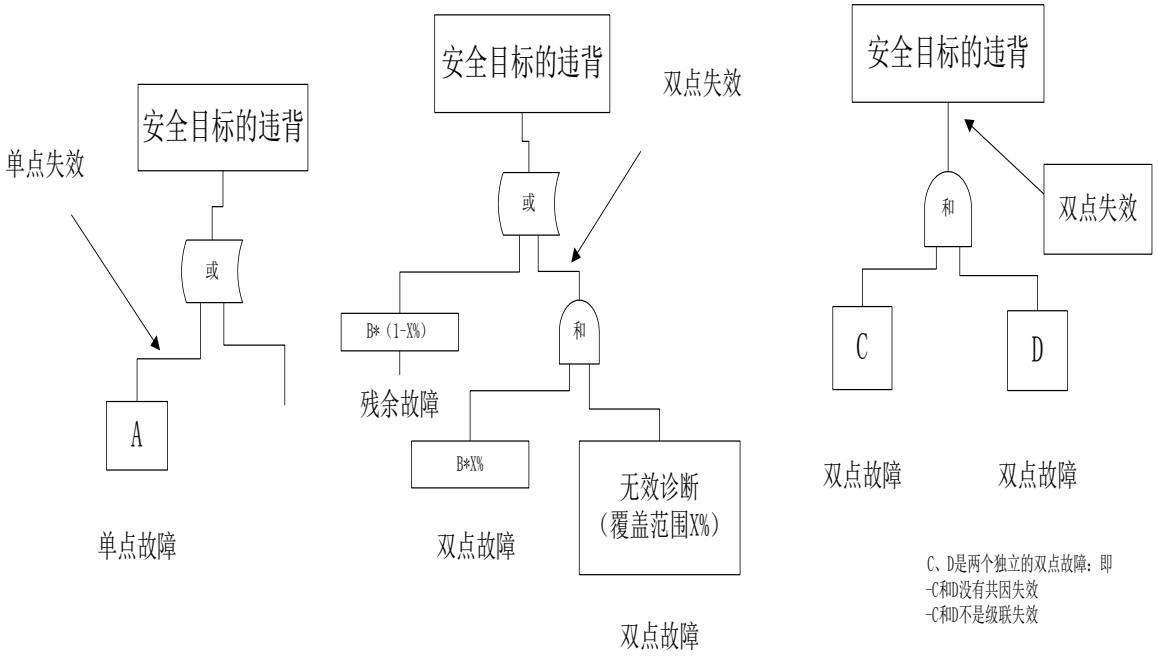


图 B.4 单点、残余和双点故障的 FTA 图解

B.3.2 构建故障树分支的示例

作为构建一个分支的示例，对寄存器R0的故障树分支进行详细描述。图B.9展示了表A.5的前两行通过或门连接。这假设寄存器R0的永久失效和瞬态失效是独立的失效模式。既然瞬态失效率和诊断覆盖率已知，那么将瞬态故障以和永久性故障相同的方式包含在故障树中。如果失效率和诊断覆盖率未知，可按照GB/T XXXX-5，8.4.7 注2的描述单独处理瞬态故障。.

注：下述示例展示了一种将永久性故障和瞬态故障结合的方法。按照GB/T XXXX-5, 8.4.7 注2，当显示瞬态故障相关时，将其包含在分析中。因在此示例中，瞬态故障既相关又可量化，所以将其以永久性故障相似的方式包含在故障树中。

首先构建瞬态故障的示例，这个分支包含一个失效率为0.032005FIT (即3.2005×10^{-11}/h)的瞬态故障，连接到与门，与门的另一个输入是一个或门。或门有两部分组成：一个是代表失效模式覆盖率为60%的确定概率（1-失效模式覆盖率，代表了对安全目标的违背），另一个是安全机制SM1的潜伏故障失效率。在事件框下面的“r=”代表了每小时的失效率，“Q=”指模块或分支在系统整个期望的生命周期中失效的概率。

注：相比于分支——诊断覆盖率与（加上）潜伏故障，确定的概率事件占主导地位。在实际系统中，分支——诊断覆盖率与（加上）潜伏故障（图B.9中19门的输出）可被忽略以简化故障树。然而，仍需评估潜伏故障度量并满足要求。

潜伏故障可导致系统无法探测出被诊断覆盖率包含且通常可被探测到的瞬态失效。需注意潜伏故障和主要故障的结合次序。如果安全机制的潜伏故障先于主要故障发生，那么主要故障不能被探测到，并且发生的危害不能得到减轻。通过带框的“L”表明该故障必须在与其相连的与门下面的其它故障发生后再发生。

SM1模块的构建来源于表A.5中安全机制部分的前两行，即探测逻辑和报警发生器的永久性失效。瞬态失效与主要故障同时发生的概率很低而可被忽略，所以瞬态失效不会引起潜伏故障，进而分析中不包含瞬态失效。这与GB/T XXXXX-5,8.4.7注2是一致的，仅在单点故障度量时考虑瞬态故障。探测逻辑的失效率为0.0029FIT。

探测逻辑的失效进一步被安全机制SM2所监控，其诊断覆盖率为90%，该事件框通过与门和SM1门相连，SM1门计算了SM1的潜伏故障概率。安全机制SM2在每次点火时运行，对SM1的潜伏故障进行探测。依照GB/T XXXXX-5，9.4.2.3，一次车辆运行的平均时间可考虑为1小时，由“DL LATENT 1”事件下的tau=1表示，再与SM2的潜伏故障覆盖率0.9（90%）相乘。报警生成器中未被SM2覆盖的故障比例，由标准失效率事件“故障生成器潜伏故障”乘以0.1（10%=1-潜伏故障覆盖率）表示。

永久性故障分支的构建采用相似的方式。三角形状的SM1事件的下划线表示其并非简单复制故障树中瞬态故障部分的已有SM1框，但却有着完全相同的失效模式。这可为共模式失效分析提供帮助，例如：在图B.9中，与门事件框“TRANS R0”和“PERM R0”包含了一个共同的分支。

表A.5中的示例包含了安全故障。考虑安全故障时，需对底事件的失效率进行调整。例如，将ALU瞬态故障（表A.5第10行，20%的安全故障）的0.00038FIT乘以0.8（1-20%），即：3.8×10^{-13}/h $\times0.8 = 3.04 \times10^{-13}$/h。

B.4 使用故障树进行概率分析

将组件失效的典型定量数据记录为失效率。对于一个具有许多“与”和“或”的复杂故障树，不能将单一失效率合并为系统总体失效率。例如，一个系统由两个通过与门连接的事件框组成，两个事件框的失效率λ1和λ2成指数分布，值非常小，（两个λxt的值都非常小，其中t代表系统生命周期），那么近似的失效概率为λ1×t × λ2xt或 λ1 λ2t^2。

如果系统是一个ASIL D的系统，使用GB/T XXXXX-5 表7中的失效率目标，每小时的目标失效率为<10^{-8}/h，该目标使用了与$\lambda_1\lambda_2t^2$不同的时间期限。一个潜在的处理方法是将每小时的目标失效率10^{-8}/h用于系统的生命周期，即10^{-8}/h×t，同时确保$\lambda_1\lambda_2t^2\leqslant10^{-8}$/h×t 或$\lambda_1\lambda_2t\leqslant10^{-8}$/h。这种方法要求了解系统生命周期，可从过去的使用情况或系统要求中获得。本附录中创建的故障树假设系统生命周期为5000小时。

对于“R0 TRANSIENT”分支，漏探测的概率主要源于诊断覆盖率（Q=0.6）的不足，并非由于潜伏故障（Q=2.175×10^{-9}）。绝大部分的实际系统都是如此，除非诊断覆盖率等于或十分接近于 100%。对于潜伏故障概率可被忽略的情况，将其从 FTA 分析中移除，以大大简化分析。潜伏故障度量仍需作为安全案例中一个单独的部分而进行论述。

GB/T XXXXX-5，9.4.3的方法中，在硬件元器件层面对安全相关故障的概率进行评估。如果分析采用了故障树的方法，则需要以能支持硬件元器件层面评估的方式构建故障树。

此示例在图B.6中体现。GB/T XXXXX-5，9.4.3.5 和 9.4.3.6以硬件元器件失效率等级及其诊断覆盖率的关联形式给出要求。如果评估是在FTA中完成的，将GB/T XXXXX-5，9.4.3.5 和 9.4.3.6的要求转化为关于硬件元器件残余失效率或单点失效率的有效要求是有益的。例如，如果选择了失效率等级1，将ASIL D目标除以100，同时选择ASIL D目标为<10^{-8} /h，GB/T XXXXX-5，9.4.3.6的要求可被转化为$\lambda_{RF,\ HW\ part} \leqslant10^{-10}$ /h，其中$\lambda_{RF,\ HW\ part}$表示硬件元器件的残余失效率。在图B.6中，我们看到在5000小时的生命周期里，CPU失效导致潜在违背安全目标概率为1.040631×10^{-6}，这相当于概率2.08×10^{-10}/h。因为这个值大于所要求的残余失效率，所以不满足9.4.3.6的要求。

依照所描述的方法使用FTA来评估对GB/T XXXXX-5，9.4.3.5或9.4.3.6的符合情况，是保守的，这是因为实际的要求是针对硬件元器件的残余失效率或单点失效率，而故障树得出的概率也包含了双点失效的场景。

因FTA不处理GB/T XXXXX-5，9.4.3.11的要求，所以有必要开展额外的分析，为满足关于双点失效场景的要求提供证据。

如果给出理由，失效率等级评级可除以一个小于100的数字。但在这种情况下，需确保同时考虑了单点故障、残余故障和更高程度的割集时，保持了正确的评级。

示例：理由可基于最小割集的数量。

B.5 故障树示例

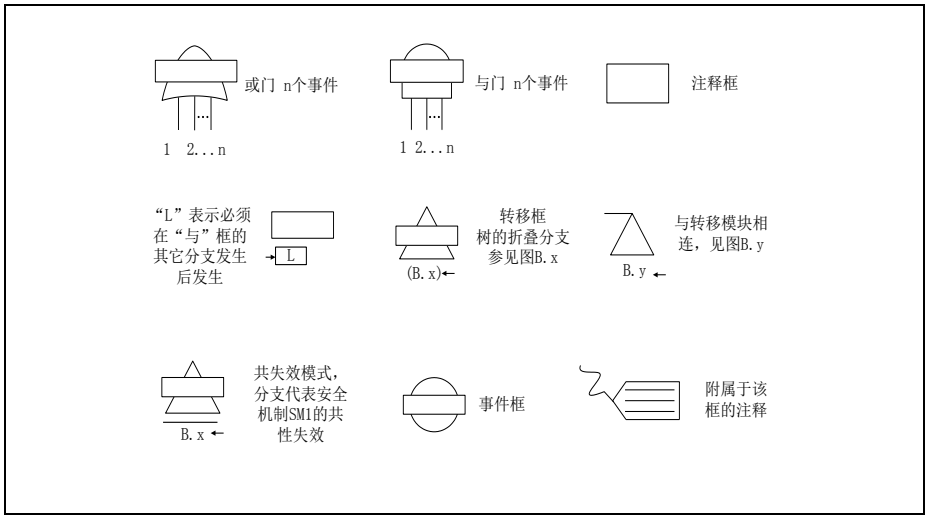


图 B.5 FTA 符号介绍说明

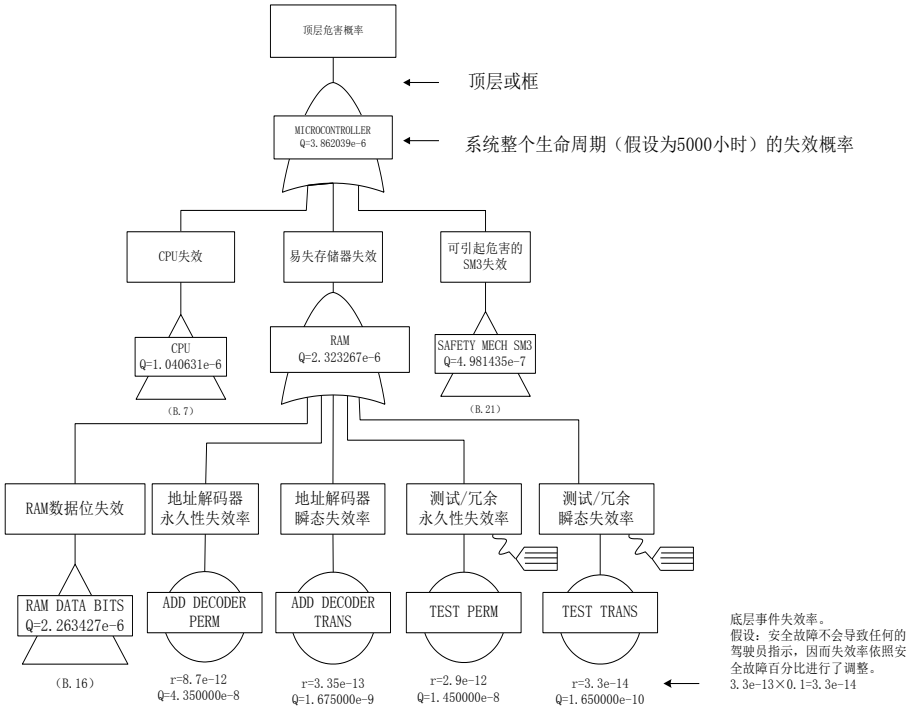


图 B.6 故障树的顶层

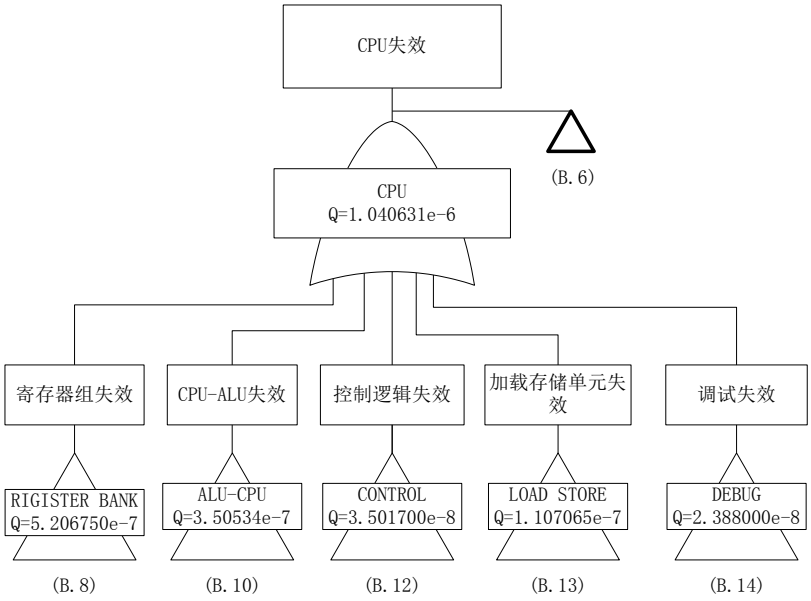


图 B.7 CPU 分支的顶层故障树

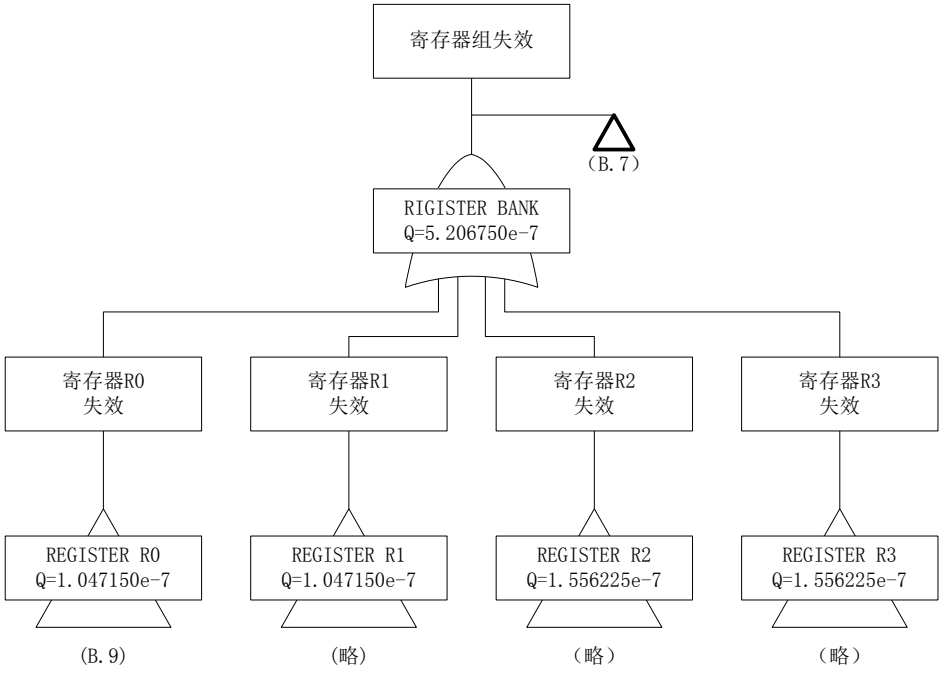


图 B.8 CPU 分支寄存器组的顶层故障树

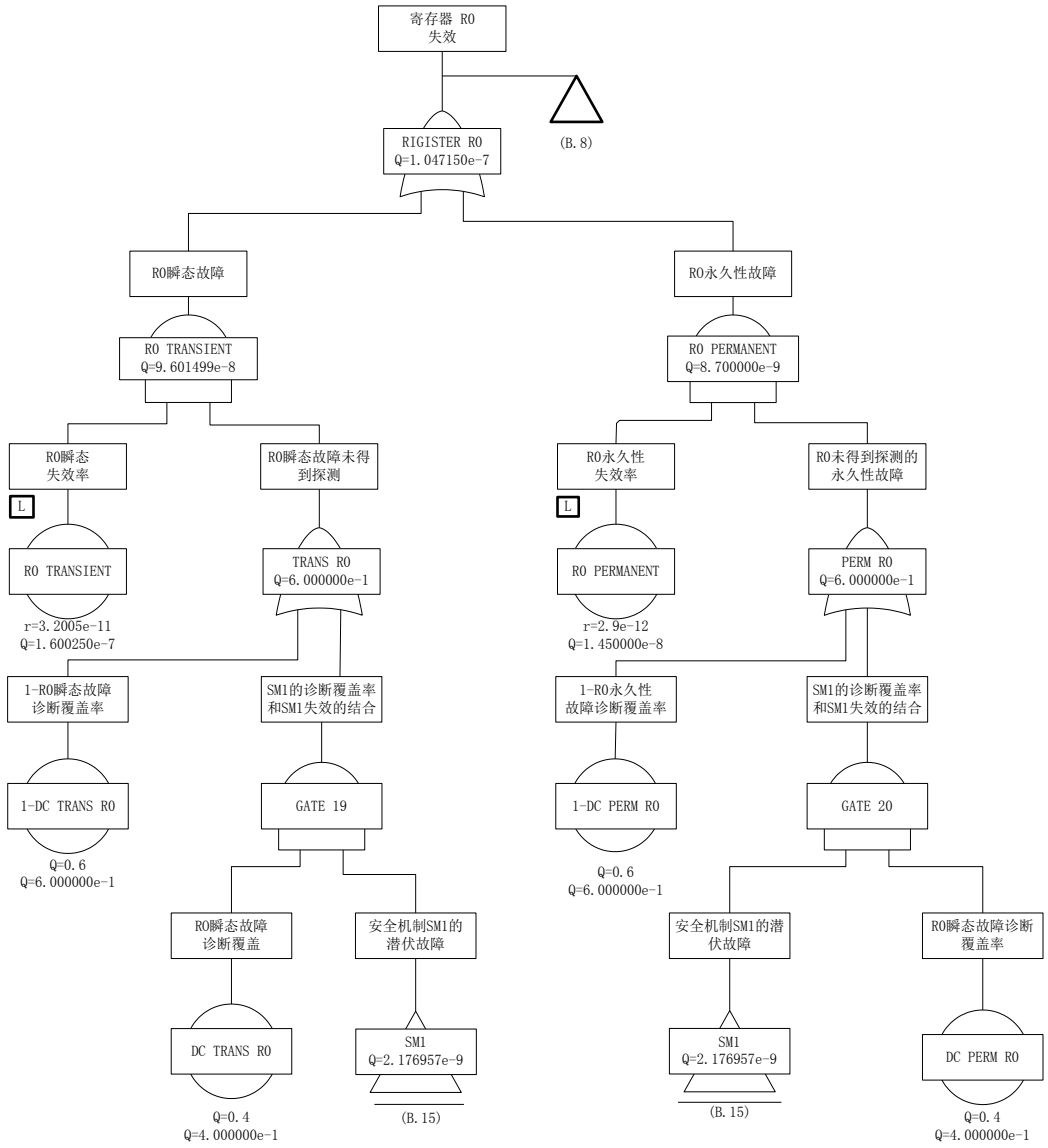


图 B.9 寄存器 R0 故障树分支

除了诊断覆盖值，图B.9是每个寄存器故障树的代表。寄存器1、2和3的详细故障树不再给出。

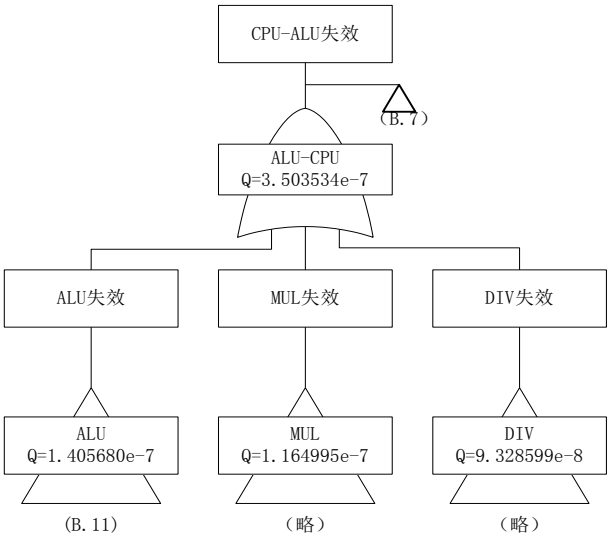


图 B.10 CPU-ALU 分支的顶层故障树

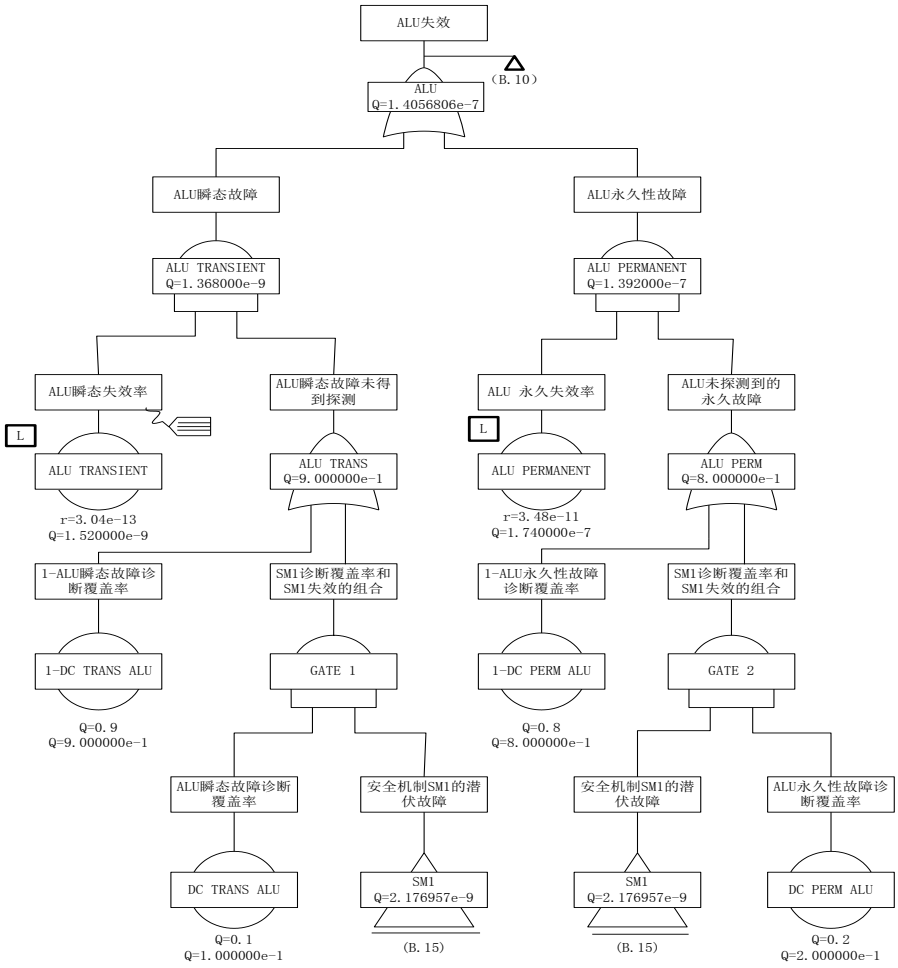


图 B.11 ALU 故障树分支

除了诊断覆盖值和FIT率，图B.11是MUL、DIV、流水线、定序器、堆控制、地址生成、加载和存储单元的故障树代表。这些分支的详细故障树不再给出。

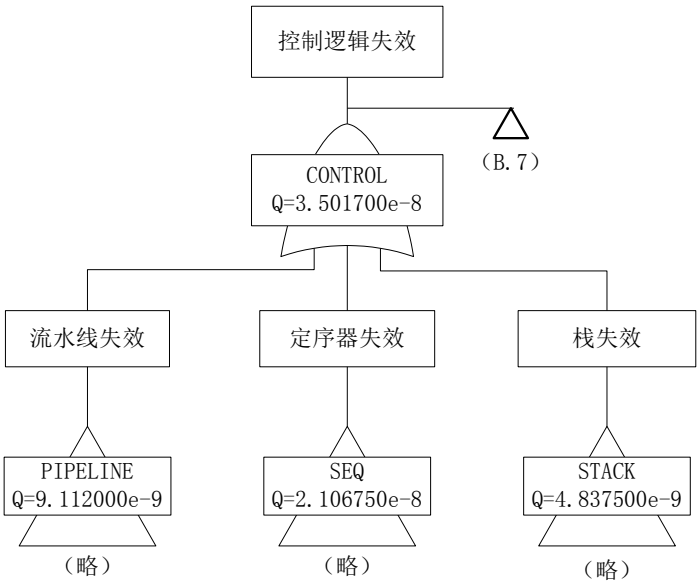


图 B.12 控制逻辑分支的顶层故障树

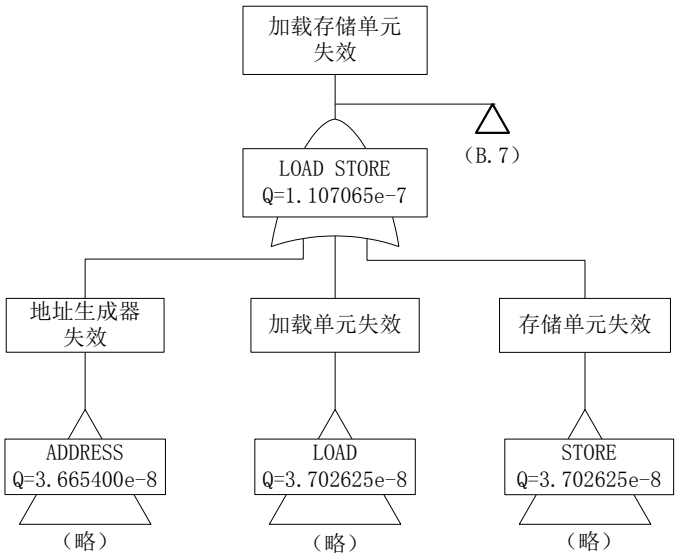


图 B.13 加载存储单元分支的顶层故障树

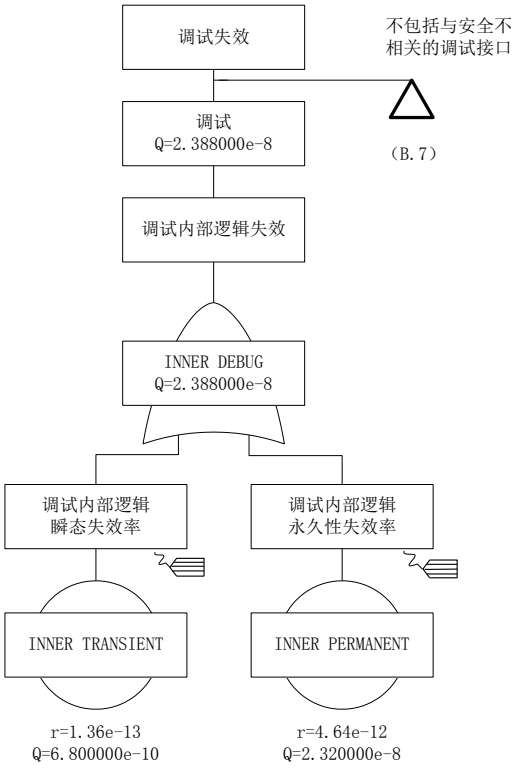


图 B.14 调试故障树分支

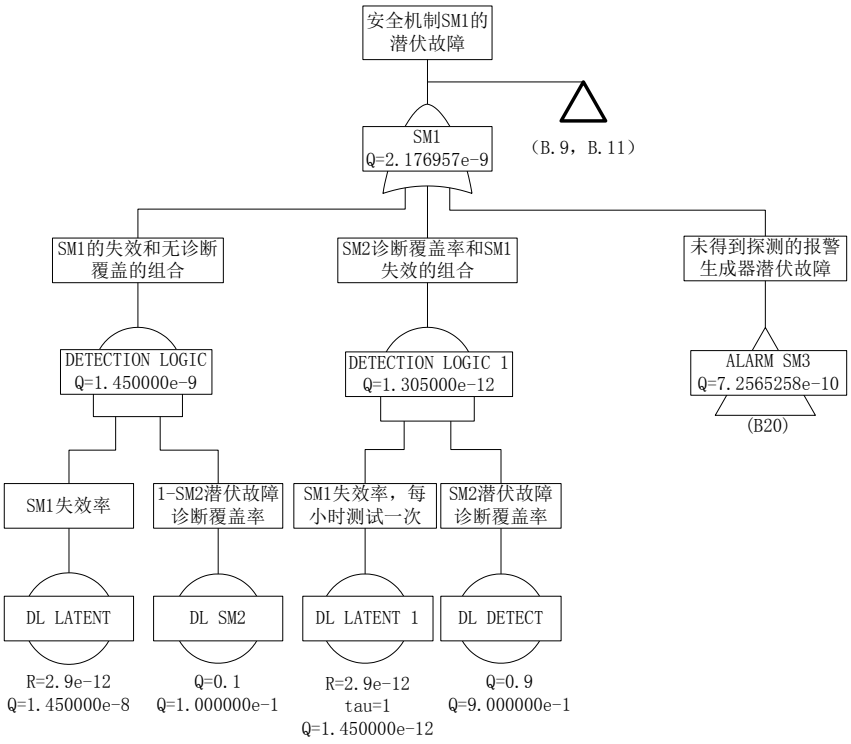


图 B.15 安全机制 SM2 对潜伏故障的覆盖

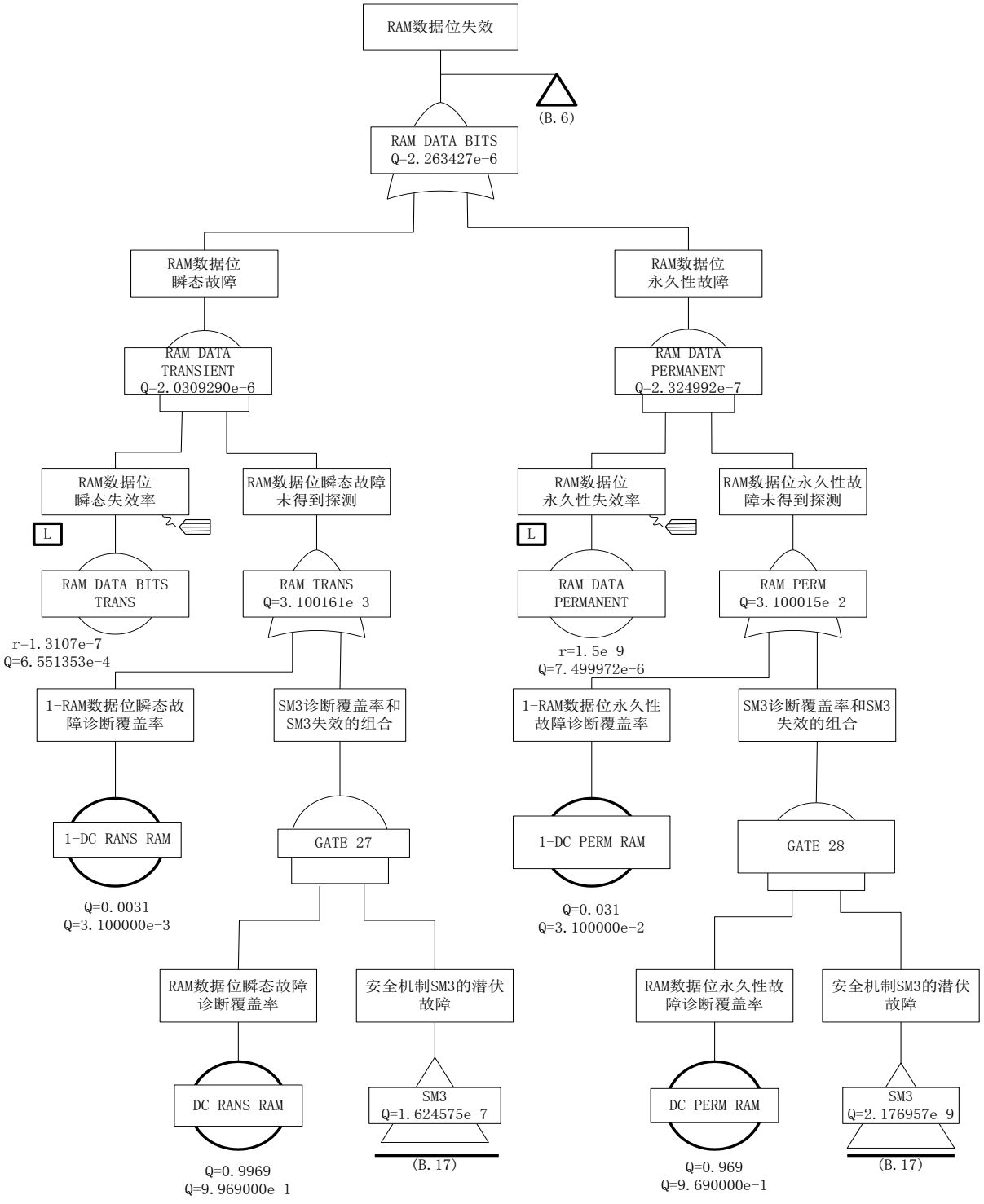


图 B.16 易失性存储器分支的顶层故障树

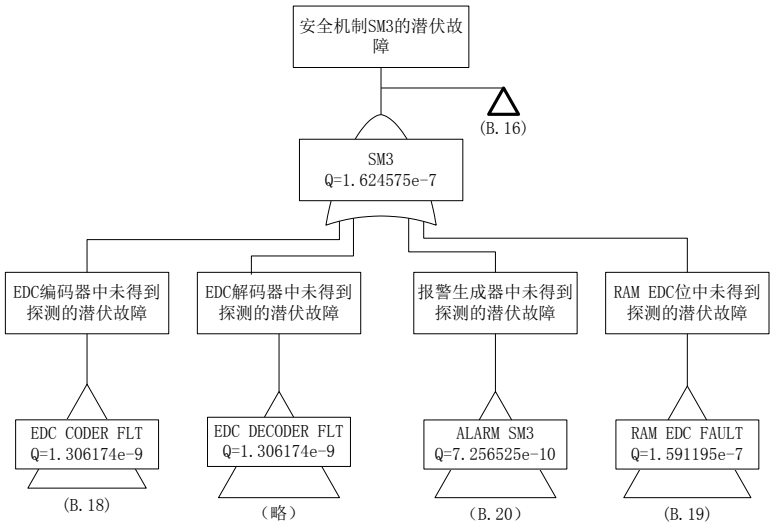


图 B.17 安全机制 SM3 分支的顶层故障树

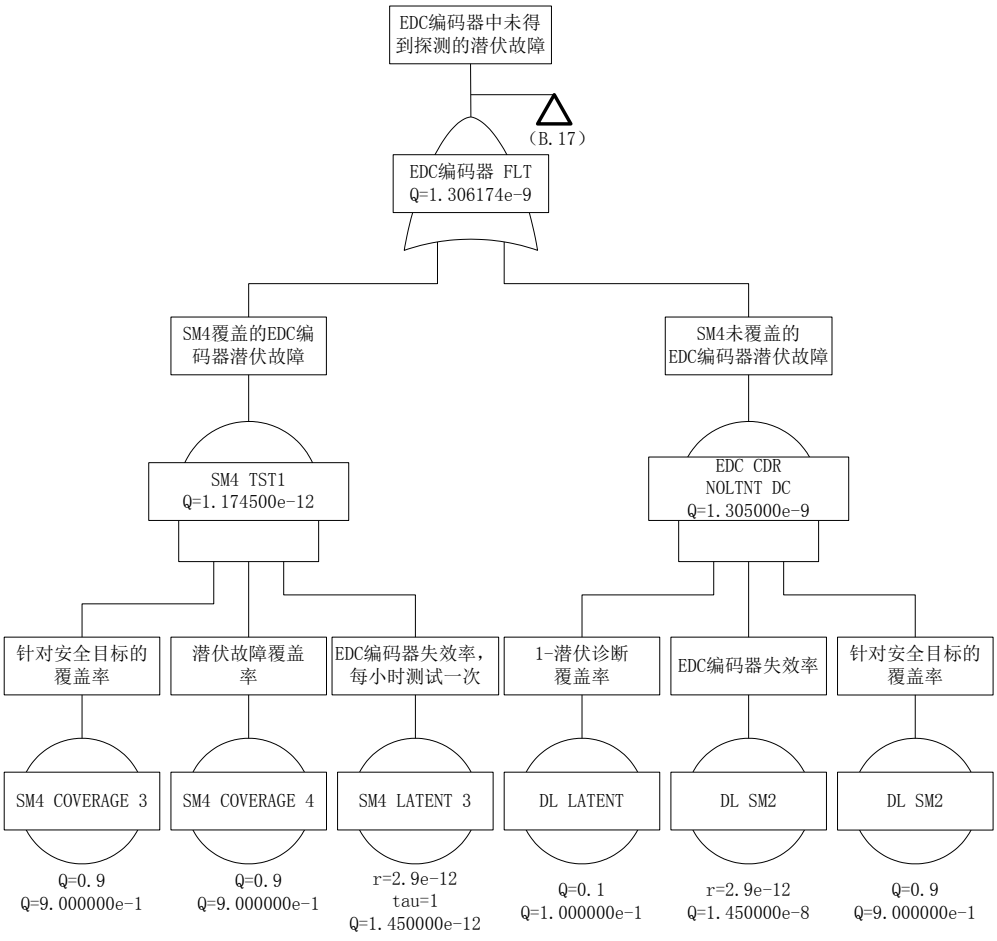


图 B.18 EDC 编码器潜伏故障树分支

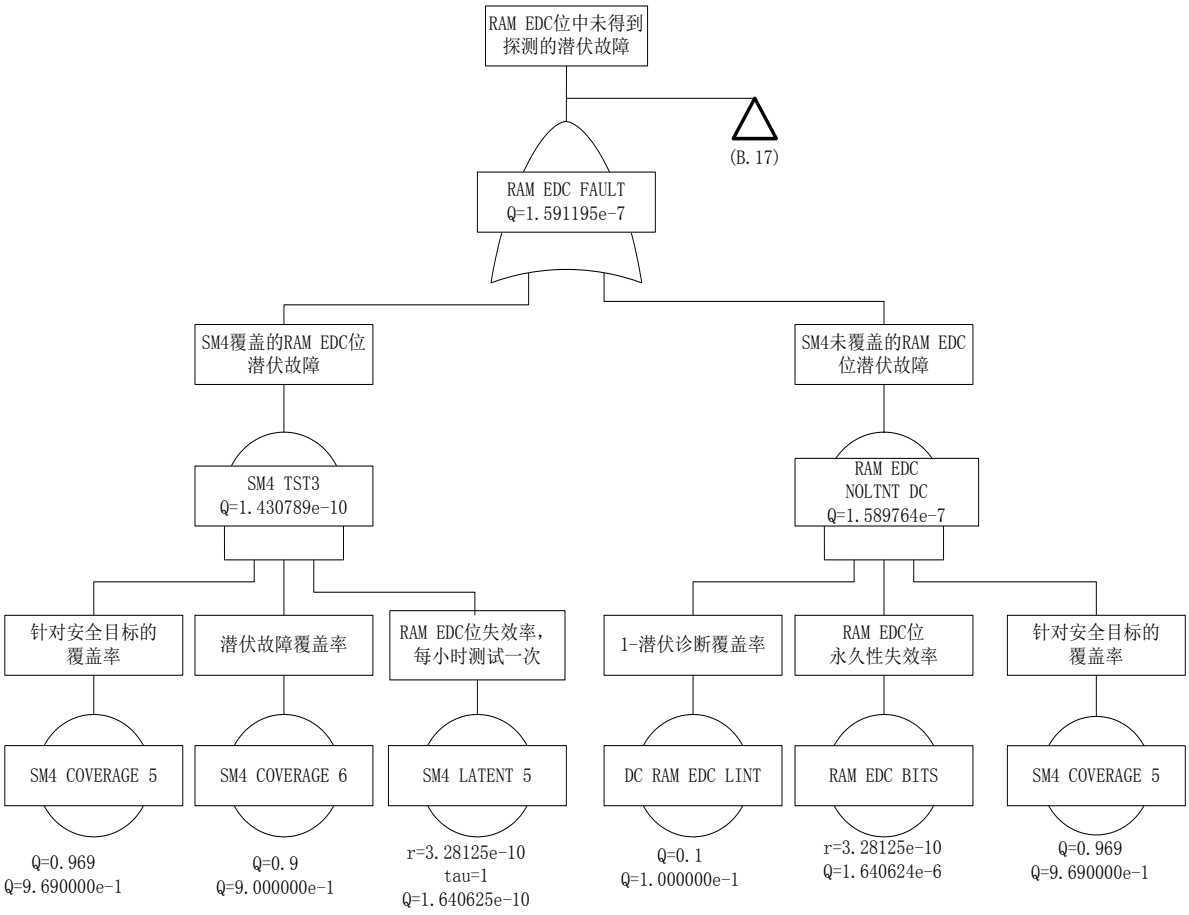


图 B.19 RAM EDC 位潜伏故障树分支

除了诊断覆盖率值，图B.19是EDC解码器的故障树代表。对该分支的详细故障树不再给出。

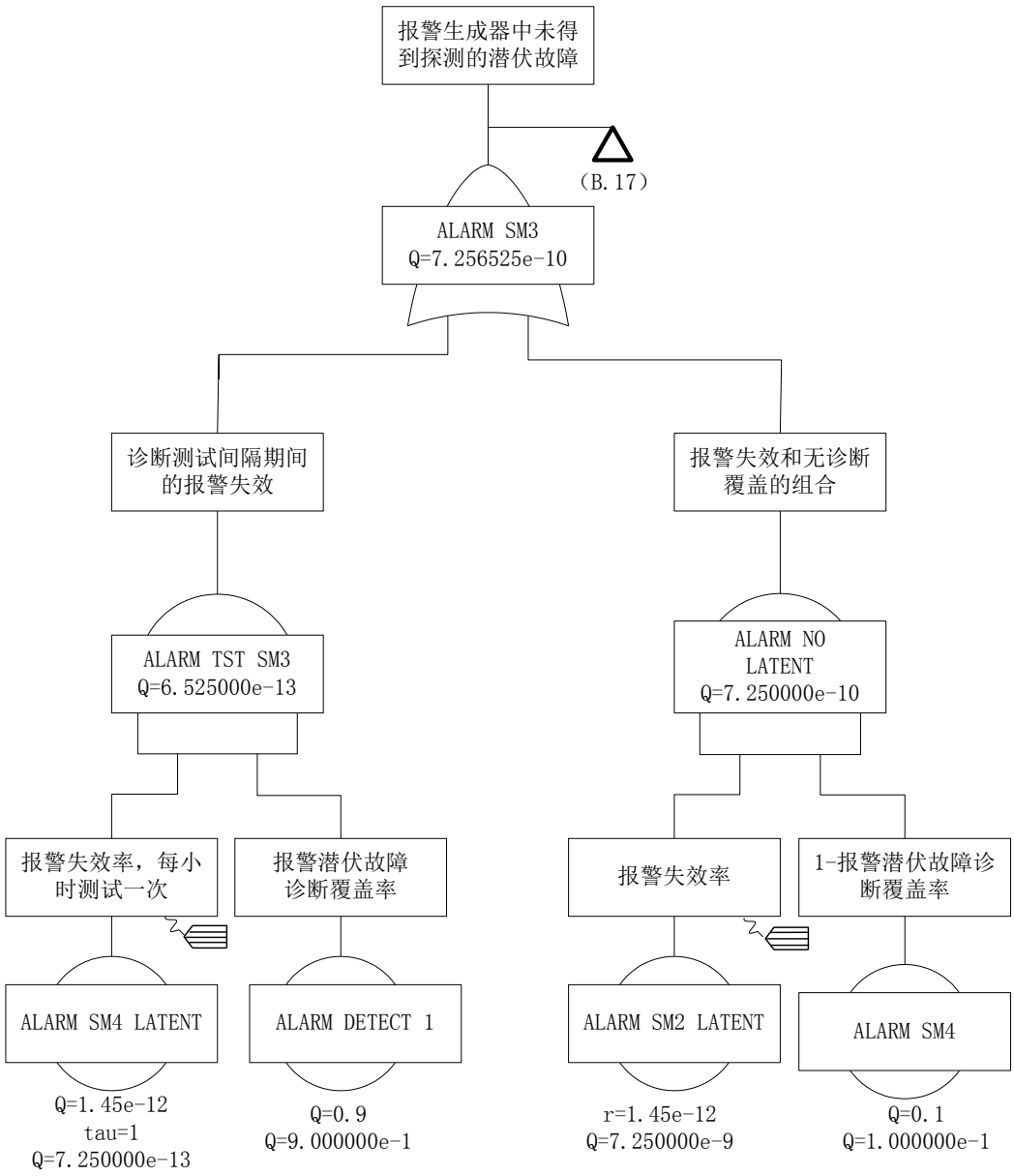


图 B.20 警报生成器潜伏故障分支

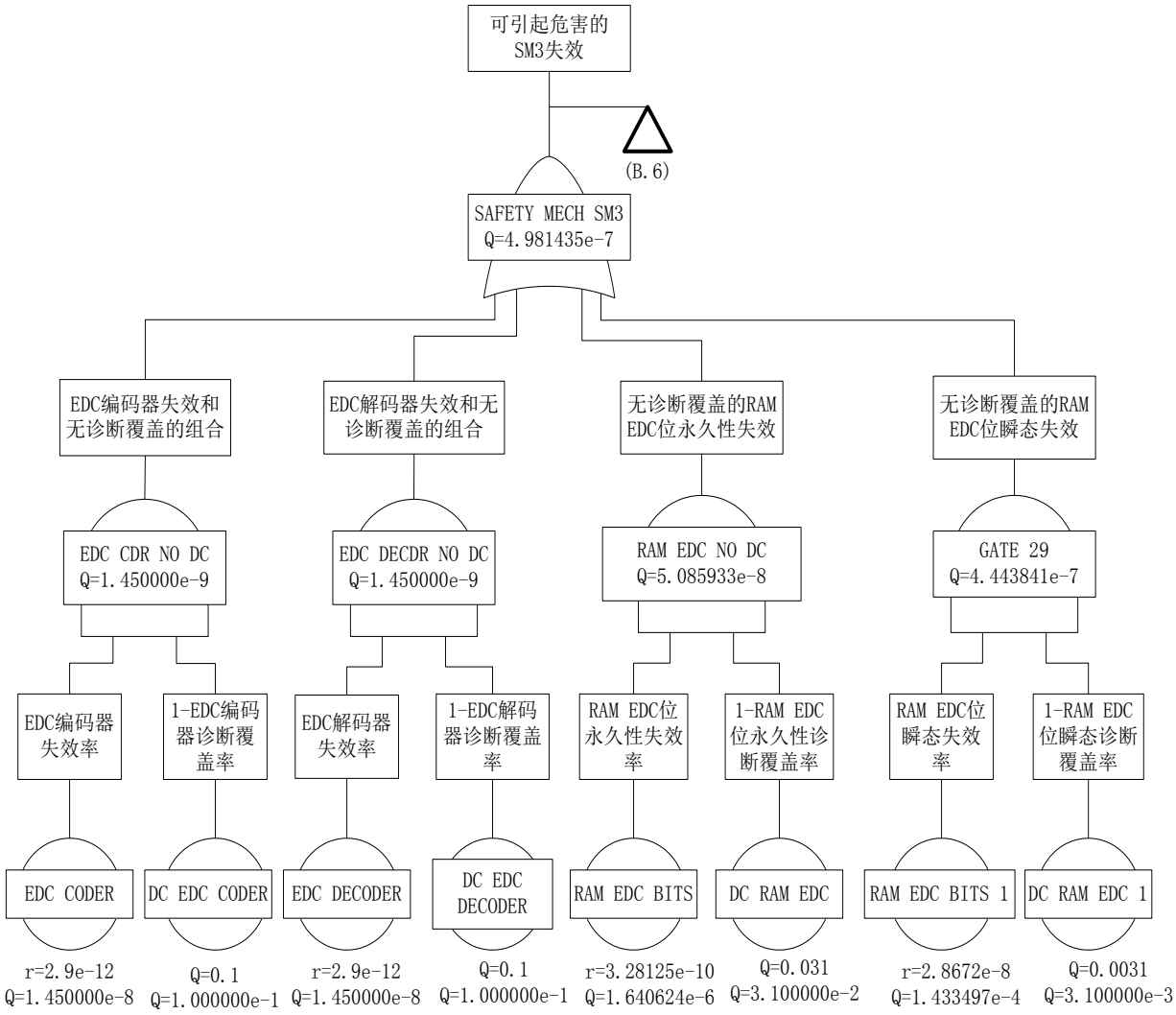


图 B.21 SM3 故障树分支，直接导致顶层危害的失效