ICS

T

中 华 人 民 共 和 国 国 家 标 准

GB/T XXXXX—XXXX

道路车辆 功能安全

第 6 部分：产品开发：软件层面

Road vehicles — Functional safety — Part 6: Product development at the software level

（征求意见稿）

2016.01.15

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：以汽车安全完整性等级为导向和以安全为导向的分析；

——第10部分：指南。

本部分为GB/T XXXXX的第6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 26262-6：2011 《道路车辆 功能安全 第6部分：产品开发：软件层面》（英文版）。

本部分的附录A、B、D为资料性附录,附录C为规范性附录。

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，新的功能越来越多地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基于其它技术的与安全相关系统的框架。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；

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和工作成果相互关联。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 GB/TXXXX-2 的第六章

图1 GB/T XXXXX概览

道路车辆 功能安全
第6部分：产品开发：软件层面

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。

本部分规定了车辆在产品开发软件层面的要求，包括：

— 启动产品开发软件层面的要求，

— 软件安全要求的定义，

— 软件架构设计，

— 软件单元设计及实现，

— 软件单元测试，

— 软件集成和测试，及

— 软件要求的验证。

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-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-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 要求

4.1 一般要求

如声明满足GB/T XXXXX的要求时，应满足每一个要求，除非有下列情况之一：

a) 按照 GB/T XXXXX-2 的要求，已经计划安全活动的剪裁并表明这些要求不适用，或，

b) 不满足要求的理由存在且可接受的，并且按照 GB/T XXXXX-2 的要求对该理由进行了评估。

标有“注”或“示例”的信息仅用于辅助理解或阐明相关要求，不应作为要求本身且不具备完备性。

将安全活动的结果作为工作成果。上一阶段工作成果作为“前提条件”的这一信息应具备。如果条款的某些要求是依照ASIL定义的或可剪裁的，某些工作成果可不作为前提条件。

“支持信息”是可供参考的信息，但在某些案例中，GB/T XXXXX不要求其作为上一阶段的工作成果，并且可以是由不同于负责功能安全活动的人员或组织等外部资源提供的信息。

4.2 对表格的诠释

表属于规范性表还是资料性表取决于上下文。在实现满足相关要求时，表中列出的不同方法有助于置信度水平。表中的每个方法是：

a) 一个连续的条目（在最左侧栏以顺序号标明，如 1,2,3），或

b) 一个选择的条目（在最左侧栏以数字后加字母标明，如 2a，2b，2c）。

对于连续的条目，全部方法应按照ASIL等级推荐予以使用。除了所列出的方法外，如果应用所列出方法以外的其它方法，应给出满足相关要求的理由。

对于选择性的条目，按照ASIL等级指示的要求，应采用适当的方法组合，不依赖于组合的方法是否在表中列出。如果所列出的方法对于一个ASIL等级来说具有不同的推荐等级，则应采用具有更高推荐等级的方法。应给出组合方法满足相关要求的理由。

注:在表中所列出的方法的理由是充分的。但是，这并不意味着有偏袒或对未列到表中的方法表示反对。

对于每种方法，应用相关方法的推荐等级取决于ASIL等级，分类如下：

— “++”表示对于指定的 ASIL 等级，高度推荐该方法；

— “+” 表示对于指定的 ASIL 等级，推荐该方法；

— “o” 表示对于指定的 ASIL 等级，未推荐或反对该方法。

4.3 基于 ASIL 等级的要求和建议

若无其它说明，对于ASIL A, B, C和D等级，应满足每一子章条的要求或建议。这些要求和建议参照安全目标的ASIL等级。如果在项目开发的早期对ASIL等级完成了分解，按照GB/T XXXXX-9第5章的要求，应遵循分解后的ASIL等级。

如果GB/T XXXXX中ASIL等级在括号中给出，则对于该ASIL等级，相应的子章节应被认为是推荐而非要求。这里的括号与ASIL等级分解无关。

5 启动软件层面产品开发

5.1 目的

本子章节的目的是计划并启动软件开发子阶段的功能安全活动。

5.2 总则

软件开发的启动是一项计划活动，即按照相关项开发的范围和复杂度确定并计划软件开发中各子阶段及其支持过程（参见GB/T XXXXX-8和GB/T XXXXX-9）。通过确定适当的方法，启动软件开发各子阶段和支持过程，以满足要求及其相应的ASIL等级。这些方法得到指南和工具的支持，这些指南和工具是为每个子阶段和支持过程而确定和计划的。

注：用于软件开发的工具可包括其它非软件工具。

示例：用于测试阶段的工具。

软件开发计划包括协调系统层面（参见GB/T XXXXX-4）和硬件层面（参见GB/T XXXXX-5）的产品开发。

5.3 本章的输入

5.3.1 前提条件

应具备下列信息：

— 项目计划（细化的），按照 GB/T XXXXX-4，5.5.1；

— 安全计划（细化的），按照 GB/T XXXXX-4，5.5.2；

— 技术安全概念，按照 GB/T XXXXX-4，7.5.1；

— 系统设计规范，按照 GB/T XXXXX-4，5.5.2；及

— 相关项集成和测试计划（细化的），按照 GB/T XXXXX-4，8.5.1。

5.3.2 支持信息

可考虑下列信息：

— 可用的经鉴定的软件工具（参见 GB/T XXXXX-8 第 11 章）；

— 可用的经鉴定的软件组件（参见 GB/T XXXXX-8 第 12 章）；

— 用于建模语言和编程语言的设计和编码指南（来自外部）；

— 方法应用的指南（来自外部）；及

— 工具应用的指南（来自外部）。

5.4 要求和建议

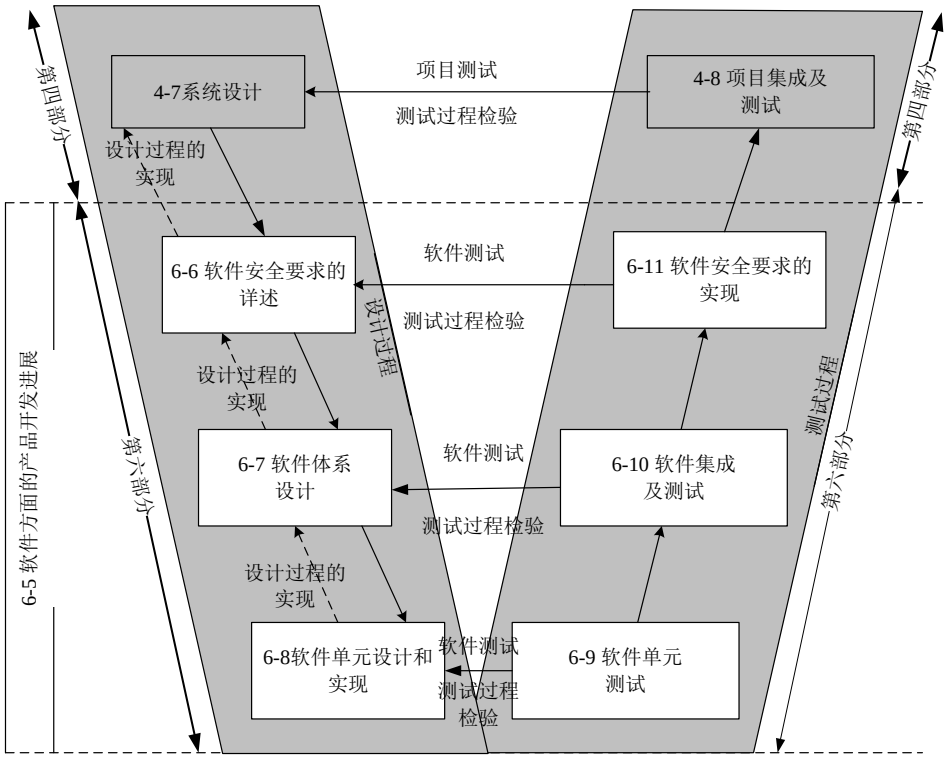
5.4.1 应计划用于软件层面产品开发的活动和确定适当的方法。

5.4.2 应按照 GB/T XXXXX-2，6.4.5，并基于图 2 给出的参考阶段模型，为软件层面产品开发进行生命周期的剪裁。

5.4.3 如果开发可配置软件，应按照附录 C。

5.4.4 相关项软件开发流程包括生命周期阶段、方法、语言和工具，应在软件生命周期的所有子阶段保持一致，并与系统和硬件开发阶段兼容，由此，所需的数据可被正确的转化。

注：相关项软件的各阶段、任务和活动（包括迭代步骤）的顺序，用于确保软件相关工作成果与硬件层面的产品开发（参见 GB/T XXXXX-5）及系统层面的产品开发（参见 GB/T XXXXX-4）保持一致。



注：图中 GB/T XXXXX 每部分的特定章节用以下方式标示：“m-n”，“m”代表部分号，“n”代表章节号，例如“4-7”代表 GB/T XXXXX-4 的第 7 章。

图 2 软件开发参考阶段模型

5.4.5 应为软件开发的每个子阶段进行如下选择，包括其应用的指南：

a) 方法；和
b) 相应的工具。

5.4.6 当选择一种合适的建模语言或编程语言时，应考虑准则：

a) 无歧义的定义；

示例：语言的语法和语义。

b) 为嵌入式实时软件和运行时错误处理提供的支持；及
c) 为模块化、抽象化和结构化的构建提供的支持；

对于通过语言本身不能充分说明的准则应由相应的指南或开发环境来覆盖。

注 1：所选择的编程语言（如 ADA、C、C++、Java、汇编或图形化的建模语言）支持 5.4.7 中给出的论题。可应用编程指南或建模指南以满足这些通则。

注 2：汇编语言可用于那些不适合使用高级语言的软件部分，如与硬件接口的底层软件、中断处理程序、或对时间敏感的算法。

5.4.7 为了支持设计和实现的正确性，建模语言或编程语言的设计和编码指南应按照表 1 所列出的通则。

注 1：对不同的编程语言，编码指南通常是不同的。

注 2：对基于模型的开发，编码指南可以是不同的。

注 3：对特定相关项的开发，可对现有编码指南进行修改。

示例：MISRA C[3] 和 MISRA AC AGC[4]是 C 语言的编码指南。

表 1 建模和编码指南覆盖的通则

通则	ASIL			
	A	B	C	D

1a	强制低复杂性[a]	++	++	++	++
1b	语言子集的使用[b]	++	++	++	++
1c	强制强类型[c]	++	++	++	++
1d	防御式实现技术的使用	o	+	++	++
1e	已建立的设计原理的使用	+	+	+	++
1f	无歧义图形化表示的使用	+	++	++	++
1g	风格指南的使用	+	++	++	++
1h	命名惯例的使用	++	++	++	++

[a] 可能需要将该通则与 GB/T XXXXX 此部分中其它方法进行恰当的折中。

[b] 方法 1b 的目的是：

— 避免使用定义模糊的语言结构，不同的模型、编码器、代码生成器或编译器可能导致对语言结构的不同解释。

— 避免容易导致错误的、来自经验的语言结构，例如：局部和全局变量的条件分配或相同命名。

— 避免可导致无法处理的运行时错误的语言结构。

[c] 方法 1c 的目的是利用在语言中非固有的强类型的原理。

5.5 工作成果

5.5.1 安全计划（细化的），由 5.4.1-5.4.7 的要求得出。

5.5.2 软件验证计划，由 5.4.1-5.4.5 和 5.4.7 的要求得出。

5.5.3 模型语言和编程语言的设计和编码指南，由 5.4.6 和 5.4.7 的要求得出。

5.5.4 工具应用指南，由 5.4.5 和 5.4.6 的要求得出。

6 软件安全要求的定义

6.1 目的

该子阶段的第一个目的是定义软件安全要求。软件安全要求来源于技术安全概念和系统设计规范。

第二个目的是对GB/T XXXXX-4第7章建立的软硬件接口要求进行细化。

第三个目的是验证软件安全要求和软硬件接口要求与技术安全概念和系统设计规范的一致性。

6.2 总则

应在GB/T XXXXX-4,第7章定义的系统设计阶段中将技术安全要求细化并分配给硬件和软件。软件安全要求的定义考虑硬件约束及其对软件的影响。该子阶段包括软件安全要求的定义，以支持后续设计阶段。

6.3 本章的输入

6.3.1 前提条件

应具备下列信息：

— 技术安全概念，按照 GB/T XXXXX-4,7.5.1；

—系统设计规范，按照 GB/T XXXXX-4,7.5.2；

— 软硬件接口规范，按照 GB/T XXXXX-4,7.5.3；

— 软件验证计划，按照 5.5.2。

6.3.2 支持信息

可考虑下列信息：

— 硬件设计规范(参见 GB/T XXXXX-5, 7.5.1);

— 方法的应用指南(来自外部)。

6.4 要求和建议

6.4.1 软件安全要求应针对每个基于软件的功能，这些功能的失效可能导致违背分配到软件的技术安全要求。

示例：失效可能导致违背安全要求的功能可以是：

— 使系统达到或维持安全状态的功能；

— 与安全相关硬件要素的故障探测、指示和处理相关的功能；

— 与软件自身故障的探测、通知和减轻相关的功能；

注 1：这些功能包括操作系统中软件的自身监控及应用层特定的软件自身监控，以探测、指示和处理应用软件中的系统性故障。

— 与车载测试和非车载测试相关功能；

注 2：车载测试可以在车辆运行过程、预运行阶段和运行后阶段内，由系统自身或通过车载网络内的其它系统执行。

注 3：非车载测试参照生产或维护阶段对安全相关的功能或属性的测试。

— 允许在生产和维护过程中对软件进行修改的功能；

— 与性能或对时间敏感的运行相关的功能。

6.4.2 软件安全要求的定义应由符合 GB/T XXXXX-4, 7.4.1 和 7.4.5 的技术安全概念和系统设计得出，并应考虑：

a) 安全要求的定义和管理，按照 GB/T XXXXX-8, 第六章；
b) 已定义的系统和硬件的配置；

示例 1：配置参数可包括增益控制、带通频率和时钟分频。

c) 软硬件接口规范；
d) 硬件设计规范的相关要求；
e) 时间约束；

示例 2：由系统层面要求的响应时间得出执行或反应时间。

f) 外部接口；及

示例 3：通讯和用户接口。

g) 对软件有影响的车辆、系统或者硬件的每个运行模式。

示例 4：硬件装置的运行模式可包括默认模式、初始化模式、测试模式和高级模式。

6.4.3 如果对软件安全要求进行了 ASIL 分解，应满足 GB/T XXXXX-9, 第五章的要求。

6.4.4 启动于 GB/T XXXXX-4, 第七章的软硬件接口规范，应细化到可以正确控制和使用硬件的程度，并应描述硬件和软件间每个与安全相关的依赖性。

6.4.5 除与 6.4.1 定义的安全要求相关的功能外，如果嵌入式软件执行了其它功能，则应对这些功能进行定义，或参考其规范。

6.4.6 应按照 GB/T XXXXX-8 第 9 章，计划对软件安全要求和细化的软硬件接口规范的验证。

6.4.7 应由负责系统开发、硬件开发和软件开发的人员共同验证细化的软硬件接口规范。

6.4.8 应按照 GB/T XXXXX-8 第 6 章和第 9 章，对软件安全要求和细化的软硬件接口要求进行验证，以展示它们：

a) 与技术安全要求的符合性和一致性；
b) 与系统设计的符合性；及
c) 与软硬件接口的一致性。

6.5 工作成果.

6.5.1 **软件安全需求规范**，由 6.4.1 到 6.4.3 及 6.4.5 的要求得出。

6.5.2 **软硬件接口规范（细化的）**，由 6.4.4 的要求得出。

注：该工作成果参照 GB/T XXXXX-5, 6.5.2 相同的工作成果。

6.5.3 **软件验证计划（细化的）**，由 6.4.6 的要求得出。

6.5.4 **软件验证报告**，由 6.4.7 和 6.4.8 的要求得出。

7 软件架构设计

7.1 目的

该子阶段的第一个目的是开发实现软件安全要求的软件架构设计。

该子阶段的第二个目的是验证软件架构设计。

7.2 总则

软件架构设计描述全部软件组件及其在层次结构中的交互。静态方面，如所有软件组件间的接口和数据路径；动态方面，如进程顺序和定时行为，都得到描述。

注：软件架构设计不必局限于某个微控制器或电控单元，它关系到技术安全概念和系统设计。每个微控制器的软件架构也在本章讨论。

为开发既实现软件安全要求又实现非安全要求的软件架构设计，在此子阶段，安全和非安全性要求应在同一开发过程中处理。

软件架构设计提供了实施软件安全要求和管理软件开发复杂性的方法。

7.3 本章的输入

7.3.1 前提条件

应具备下列信息：

— 安全计划（细化的），按照 5.5.1；

— 用于建模及编程语言的设计和编码指南，按照 5.5.3；

— 软硬件接口规范，按照 GB/T XXXXX-4，7.5.3；

— 软件安全需求规范，按照 6.5.1；

— 软件验证计划（细化的），按照 6.5.3；及

— 软件验证报告，按照 6.5.4。

7.3.2 支持信息

可考虑下列信息：

— 技术安全概念 （见 GB/T XXXXX-4，7.5.1）；

— 系统设计规范 （见 GB/T XXXXX-4，7.5.2）；

— 可用的、经鉴定的软件组件 （见 GB/T XXXXX-8，第 12 章）；

— 工具使用指南，按照 5.5.4；及

— 应用方法指南（来自外部）。.

7.4 要求和建议

7.4.1 为确保软件架构设计获取必要信息以允许后续开发活动得到正确且有效的执行，应使用表 2 中列出的软件架构设计的标记法，对软件架构设计进行恰当抽象层级的描述。

表 2 软件架构设计的标记法

方法		ASIL			
		A	B	C	D
1a	非正式标记法	++	++	+	+
1b	半正式标记法	+	++	++	++
1c	正式标记法	+	+	+	+

7.4.2 在软件架构设计开发中应考虑下述方面：

a) 软件架构设计的可验证性；

注：这表明软件架构设计和软件安全要求之间的双向可追溯性。

b) 可配置软件的适用性；

c) 软件单元设计和实现的可行性；

d) 软件集成测试中，软件架构的可测性；及

e) 软件架构设计的可维护性。

7.4.3 为避免因高度复杂性导致的失效，应通过使用表 3 列出的原则，使软件架构设计具有以下属性：

a) 模块化；

b) 封装；及

c) 简单性。

表 3 软件架构设计原则

方法		ASIL			
		A	B	C	D
1a	软件组件的层次	++	++	++	++
1b	限制软件组件的 [a]	++	++	++	++
1c	限制接口规模 [a]	+	+	+	+
1d	每个软件组件内 [b]	+	++	++	++
1e	软件组件间的耦 [a, b, c]	+	++	++	++
1f	恰当调度的特性	++	++	++	++

1g	限制中断的使用 [a, d]	+	+	+	++

[a] 在方法 1b、1c、1e 和 1g 中，“限制”表示与其它设计考虑进行平衡后的最低程度。

[b] 例如，方法 1d 和 1e 可通过分隔关注点的方法实现，这些关注点代表了对特定概念、目标、任务或目的相关的软件部分进行识别、封装和操作的能力。

[c] 方法 1e 针对软件组件的外部耦合限制。

[d] 所使用的任何中断都必须是基于优先级的。

注：基于表 3 所列方法的适当折中可能是必要的，因为这些方法不互相排斥。

7.4.4 软件架构设计应被开发到能够识别出所有软件单元的程度。

7.4.5 软件架构设计应描述：

a) 软件组件的静态设计方面；及

注 1：静态设计方面涉及：

— 软件结构包括其分级层次；

— 数据处理的逻辑顺序；

— 数据类型和它们的特征参数；

— 软件组件的外部接口；

— 软件的外部接口；及

— 约束（包括架构的范围和外部依赖）。

注 2：在基于模型开发的情况下，结构建模是整个建模活动的固有部分。

b) 软件组件的动态设计方面。

注 1：动态设计方面涉及：

— 功能性和行为；

— 控制流和并发进程；

— 软件组件间的数据流；

— 对外接口的数据流；及

— 时间的限制。

注 2：为确定动态行为（如任务、时间片和中断），需要考虑不同的运行状态（如开机、关机、正常运行、标定和诊断）。

注 3：为描述动态行为（如任务、时间片和中断），需要定义通讯关系和所分配的系统硬件（如 CPU 和通讯通道）。

7.4.6 每个与安全相关的软件组件应被归类为下述之一：

a) 新开发的；

b) 修改后使用的；或

c) 未经修改复用的。

7.4.7 新开发的或修改后使用的安全相关软件组件应按照 GB/T XXXXX 来开发。

注：在这些情况下，不适用 GB/T XXXXX-8 第 12 章。

7.4.8 应按照 GB/T XXXXX-8 第 12 章，对未经修改重用的安全相关软件组件进行鉴定。

注：使用经鉴定的软件组件不影响第 10 章和第 11 章的适用性。然而，第 8 章和第 9 章描述的某些活动可以被省略。

7.4.9 应将软件安全要求分配给软件组件。因此，每个软件组件应按照分配给它的要求中最高的 ASIL 等级来进行开发。

注：根据这一分配，进一步细化软件安全要求可能是必要的。

7.4.10 如果嵌入式软件不得不实现不同 ASIL 等级的软件组件，或实现安全相关及非安全相关的软件组件，除非软件组件符合 GB/T XXXXX-9 第 6 章定义的兼容性准则，否则全部嵌入式软件必须按照最高 ASIL 等级来处理。

7.4.11 如果用软件分区（划分、分割）（参见附录 D）实现软件组件间免于干扰，那么应该确保：

a) 共享资源的使用方式应确保软件分区免于干扰；

注 1：一个软件分区内的任务彼此之间不能免于干扰。

注 2：一个软件分区不能改变其它软件分区的代码或数据，也不能访问其它软件分区的非共享资源。

注 3：一个软件分区从共享资源获取的服务不能被另一个软件分区影响。这包括相关资源的性能，以及计划访问资源的使用率、延迟、抖动和持续时间。

b) 由专用的硬件功能或等效方法来支持软件分区（该要求适用于 ASIL D，按照 4.3）

c) 执行软件分区的软件部分，按照分配给软件分区要求的相同 ASIL 等级进行开发，或按照比分配给软件分区要求的最高 ASIL 等级更高的一个 ASIL 等级进行开发；且

注：一般来说操作系统提供或支持软件分区。

d) 在软件集成和测试（按照第 10 章）过程中执行软件分区的验证。

7.4.12 如果软件安全要求的执行依赖于软件组件间免于干扰或足够的独立性，那么应按照 GB/T XXXXX-9 第 7 章进行相关失效的分析。

7.4.13 应按照 GB/TXXXXX-9 第 8 章在软件架构层面执行安全分析，目的是：

— 识别或确认软件的安全相关部分；及

— 支持安全机制的定义和验证其有效性。

注：安全机制可被定义为同时覆盖随机硬件失效和软件故障的有关问题。

7.4.14 为了在软件架构层面定义必要的软件安全机制,应基于符合 7.4.13 的安全分析结果,使用表 4 列出的错误探测机制。

注：当分配给软件的技术安全要求没有对软件安全机制的使用进行直接要求时，那么应在系统层面对软件安全机制的使用进行评审，以分析对系统行为的潜在影响。

表 4 软件架构层面的错误探测机制

方法		ASIL			
		A	B	C	D
1a	输入输出数据的范围检查	++	++	++	++
1b	真实性检查[a]	+	+	+	++
1c	数据错误探测[b]	+	+	+	+
1d	外部监控设备[c]	o	+	+	++
1e	控制流监控	o	+	++	++

1f	多样化的软件设计	o	o	+	++

[a] 合理性检查可包括使用：期望行为的参考模型、断言检查、或不同来源的比较信号。

[b] 可被用于探测数据错误（包括检错码和多重数据存储）的方法类型。

[c] 外部监控设备可以是，例如，ASIC 或其它执行看门狗功能的软件要素。

7.4.15 此子章节适用于等级为 ASIL(A)、(B)、C 和 D 的安全目标，按照 4.3：基于符合 7.4.13 的安全分析结果，为了在软件架构层面定义必需的软件安全机制，应使用表 5 列出的错误处理机制。

注 1：当分配给软件的技术安全要求没有直接要求时，那么在系统层面对软件安全机制的使用进行评审，以分析对系统行为的潜在影响。

注 2： 在 GB/T XXXXX-5 中描述了由于硬件导致的在软件架构层面的潜在危害的分析。

表 5 软件架构层面的错误处理机制

方法		ASIL			
		A	B	C	D
1a	静态恢复机制 [a]	+	+	+	+
1b	适度降级 [b]	+	+	++	++
1c	独立并行冗余 [c]	o	o	+	++
1d	数据纠错码	+	+	+	+

[a] 静态恢复机制可包括使用恢复块、返回恢复、向前恢复以及通过备份来恢复。

[b] 在软件层面的适度降级是指区分不同功能的优先次序，从而将潜在故障对功能安全的不利影响降低到最小程度。

[c] 独立并行冗余可通过在每个并行路径的不同软件来实现。

7.4.16 如果软件架构设计所引入的新的危害没有被现有的安全目标覆盖，应按照 GB/T XXXXX-8 第 8 章所述的变更管理流程在危害分析和风险评估中对它们进行介绍和评估。

注：未体现在安全目标中的新识别危害，通常是非功能性危害。如果这些非功能性危害超出了本标准的范畴，那么建议在危害分析和风险评估中用以下声明“因不属于 GB/T XXXXX 的范畴，而未对该危害分配 ASIL”来标注它们，然而，为了参考，允许对其分配一个 ASIL 等级。

7.4.17 应该对嵌入式软件所需资源进行上限预估，包括：

a) 执行时间；
b) 存储空间；及

示例 ：用于存储堆和栈的 RAM，用于存储程序和非易失数据的 ROM。

c) 通讯资源。

7.4.18 软件架构设计应按照 GB/T XXXXX-8 第 9 章来进行验证，并使用表 6 中所列出的软件架构验证方法来论证下述属性：

a) 与软件安全要求的符合性；

b） 与目标硬件的兼容性；及

注：这包括在 7.4.17 中定义的资源。

c） 与设计指南保持一致。

表 6 软件架构设计验证方法

方法		ASIL			
		A	B	C	D
1a	设计走查[a]	++	+	o	o
1b	设计检查[a]	+	++	++	++
1c	对设计中的动态部分进行仿真[b]	+	+	+	++
1d	生成原型	o	o	+	++
1e	形式验证	o	o	+	+
1f	控制流分析[c]	+	+	++	++
1g	数据流分析[c]	+	+	++	++

[a] 在基于模型开发的情况下，这些方法可以在模型中应用。

[b] 方法 1c 要求为软件架构的动态部分使用可执行模型。

[c] 控制流和数据流分析可以限制在安全相关组件和它们的接口上。

7.5 工作成果

7.5.1 软件架构设计规范，由 7.4.1 至 7.4.6、7.4.9、7.4.10、7.4.14、7.4.15 和 7.4.17 的要求得出。

7.5.2 安全计划（细化的），由 7.4.7 的要求得出。

7.5.3 软件安全需求规范（细化的），由 7.4.9 的要求得出。

7.5.4 安全分析报告，由 7.4.13 的要求得出。

7.5.5 相关失效分析报告，由 7.4.12 的要求得出。

7.5.6 软件验证报告（细化的），由 7.4.18 的要求得出。

8 软件单元设计和实现

8.1 目的

本子阶段的第一个目的是按照软件架构设计和相关的软件安全要求，定义软件单元。

本子阶段的第二个目的是按照定义，实现软件单元。

本子阶段的第三个目的是静态验证软件单元的设计及其实现。

8.2 总则

基于软件架构设计，开发软件单元的详细设计。详细设计将分别按照建模或编码指南，以模型或直接以源代码的形式实现。在进入到软件单元测试阶段前，对详细设计和实现进行静态验证。如果使用手工开发代码，在源代码层面具备与实现相关的特性。如果使用基于模型开发的自动生成代码，这些特性用于模型而不需要用于源代码。

为了将软件安全要求和非安全相关要求同时实施到同一软件单元设计中，本子阶段用一个开发过程处理安全相关和非安全相关要求。

软件单元实现包含源代码生成和转换为目标代码。

8.3 本章的输入

8.3.1 前提条件

应具备下列信息：

— 建模和编程语言的设计及编码指南，按照 5.5.3；

— 软件验证计划（细化的），按照 6.5.3；

— 软件架构设计规范，按照 7.5.1；

— 安全计划（细化的），按照 7.5.2；

— 软件安全需求规范（细化的），按照 7.5.3；及

— 软件验证报告（细化的），按照 7.5.6。

8.3.2 支持信息

可考虑下列信息：

— 技术安全概念（参见 GB/T XXXXX-4，7.5.1）；

— 系统设计规范（参见 GB/T XXXXX-4，7.5.2）；

— 工具应用指南，按照 5.5.4；

— 软硬件接口规范（细化的）（参见 6.5.2）；

— 安全分析报告，按照 7.5.4；及

— 方法应用指南（来自外部）。

8.4 要求和建议

8.4.1 如果软件单元是安全相关的，应符合此子阶段中的要求。

注： “安全相关”是指单元实现了安全要求，或此单元不满足与其它单元的共存原则（参见 GB/T XXXXX-9，第 6 章）

8.4.2 为确保软件单元设计获得必要的信息以允许后续开发活动得到正确和有效的执行，应使用表 7 中列出的标记法描述软件单元设计。

表 7 软件单元设计的标记法

方法		ASIL			
		A	B	C	D
1a	自然语言	++	++	++	++
1b	非正式标记法	++	++	+	+
1c	半正式标记法	+	++	++	++
1d	正式标记法	+	+	+	+

注：在应用自动代码生成的基于模型开发的情况下，将软件单元设计的表示方法，用于作为代码生成基础的模型中。

8.4.3 软件单元的定义应将功能表现和内部设计描述到必要的细节程度以支持其实现。

示例：内部设计可包含对寄存器使用和数据存储的限制。

8.4.4 应运用表 8 列出的源代码层面软件单元设计和实现的设计原则，以具有如下特性：

a) 基于软件架构设计，软件单元内的子程序和函数按照正确次序执行；
b) 软件单元间接口的一致性；
c) 软件单元内和软件单元间的数据流及控制流的正确性；
d) 简单性；
e) 可读性和可理解性；
f) 鲁棒性；

示例：避免不合理值、执行错误、以零做除数、数据流及控制流错误的方法。

g) 软件修改的适宜性；及
h) 可测性。

表 8——软件单元设计和实现的设计原则

方法		ASIL			
		A	B	C	D
1a	子程序和函数采用一个入口和一个出口 [a]	++	++	++	++
1b	无动态对象或动态变量，在其产生过程中也没有在线测试 [a,b]	+	++	++	++
1c	变量初始化	++	++	++	++
1d	不能重复使用变量名称 [a]	+	++	++	++
1e	避免全局变量，否则需证明对全局变量的使用是合理的 [a]	+	+	++	++
1f	限制使用指针 [a]	o	+	+	++
1g	无隐式类型转换 [a,b]	+	++	++	++
1h	无隐藏数据流或控制流 [c]	+	++	++	++
1i	没有无条件跳转 [a,b,c]	++	++	++	++
1j	无递归	+	+	++	++

[a] 方法 1a、1b、1d、1e、1f、1g 和 1i 可能不适用于在基于模型开发中用到的图形模型标记法。

[b] 方法 1g 和 1i 不适用于汇编语言编程。

[c] 方法 1h 和 1i 通过跳转或全局变量，降低了模型数据流和控制流的可能性。

注：对于 C 语言，MISRA C 涵盖了表 8 列出的很多方法。

8.4.5 应按照 GB/T XXXXX-8 第 9 章，并用表 9 列出的验证方法，对软件单元设计和实现进行验证，以证明：

a) 对软硬件接口规范的符合性（按照 GB/T XXXXXX-5，6.4.10）；
b) 通过追溯性表明满足了分配给软件单元的软件安全要求（按照 7.4.9）；
c) 源代码与其设计规范的一致性；

注：对基于模型开发的情况，要求 c)仍然适用。

d) 源代码与其编码指南的一致性（参见 5.5.3）；及
e) 软件单元的实现与目标硬件的兼容性。

表 9 软件单元设计和实现的验证方法

方法		ASIL			
		A	B	C	D
1a	走查 [a]	++	+	o	o
1b	检查 [a]	+	++	++	++
1c	半正式验证	+	+	++	++
1d	正式验证	o	o	+	+
1e	控制流分析 [b,c]	+	+	++	++
1f	数据流分析 [b,c]	+	+	++	++
1g	静态代码分析	+	++	++	++
1h	语义代码分析 [d]	+	+	+	+

[a] 在基于模型的软件开发情况下，可在模型层面对软件单元规范的设计和开发进行验证。

[b] 方法 1e 和 1f 可用于源代码层面。这些方法同时适用于手动代码开发和基于模型的开发。

[c] 方法 1e 和 1f 可作为方法 1d、1g 或 1h 的一部分。

[d] 方法 1h 通过使用变量可能值的抽象表示，用于源代码的数学分析。为此，不一定需要翻译和执行源代码。

注：表 9 仅列出了静态验证技术。动态验证技术（例如，测试技术）在表 10、11 和 12 中给出。

8.5 工作成果

8.5.1 **软件单元设计规范**，由 8.4.2 到 8.4.4 的要求得出。

注：在基于模型开发的情况下，实现的模型和支持的说明文档用表 8 列出的技术定义软件单元。

8.5.2 **软件单元实现**，由 8.4.4 的要求得出。

8.5.3 **软件验证报告（细化的）**，由 8.4.5 的要求得出。

9 软件单元测试

9.1 目的

这个子阶段的目的是要证明，软件单元满足软件单元设计规范，且不包含非期望的功能。

9.2 总则

根据软件单元设计规范，建立软件单元测试流程，并按照该流程执行测试。

9.3 本章的输入

9.3.1 前提条件

应具备下列信息：

— 硬件与软件接口规范(细化的)，按照 6.5.2；

— 软件验证计划(细化的），按照6.5.3；

— 安全计划(细化的)，按照7.5.2；

— 软件单元设计规范，按照8.5.1；

— 软件单元实现，按照8.5.2；和

— 软件验证报告（细化的），按照8.5.3。

9.3.2 支持信息

可考虑下列信息：

— 工具应用指南，按照5.5.4；和

— 方法应用指南（来自外部）。

9.4 要求和建议

9.4.1 如果软件单元是与安全相关的，则应符合本子章节的要求。

注：“安全相关”是指单元实现安全要求，或此单元不满足与其它单元的共存原则。

9.4.2 应按照 GB/T XXXXX-8 第 9 章，计划、定义和执行软件单元测试。

注 1：基于 GB/T XXXXX-8 第 9 章的定义，软件单元测试的对象是软件单元。

注 2：基于模型的软件开发，实现模型的相应部分也代表了测试计划的对象。根据选定的软件开发过程，测试对象可以是由模型产生的代码或该模型本身。

9.4.3 应用表 10 中列出的软件单元测试方法，证明软件单元达到：

a) 符合软件单元的设计规范（按照第 8 章）；

b) 符合软硬件接口的定义（按照 GB/T XXXXX-5，6.4.10）；

c) 已定义的功能；

d) 确信没有非预期的功能；

e) 鲁棒性；和

示例：不存在不能访问的软件，错误检测的有效性和错误处理机制的有效性。

f) 足够的资源来支持它们的功能。

表 10 软件单元测试方法

方 法		ASIL			
		A	B	C	D
1a	基于需求的测试 [a]	++	++	++	++
1b	接口测试	++	++	++	++
1c	故障注入测试 [b]	+	+	+	++

1d	资源使用测试 [c]	+	+	+	++
1e	如果适用，在模型和代码之间背靠背对比测试 [d]	+	+	++	++

[a] 单元层面的软件要求是基于要求测试的基础。

[b] 这包括任意故障注入（例如通过破坏变量的值、通过引入编码突变，或通过破坏 CPU 寄存器的值）。

[c] 仅当在目标硬件上执行软件单元测试或目标处理器的仿真器支持资源使用测试时，才能恰当评估资源使用测试的某些方面。

[d] 此方法需要一种能模拟软件单元功能的模型。这里，模型和代码是以同样的方式进行触发，并将结果互相比较。

9.4.4 应使用表 11 列出的方法得到测试案例，以恰当定义符合 9.4.3 的软件单元测试的测试案例。

表 11 软件单元测试案例的得出方法

方法		ASIL			
		A	B	C	D
1a	要求分析	++	++	++	++
1b	等价类的生成和分析 [a]	+	++	++	++
1c	边界值分析 [b]	+	++	++	++
1d	错误推测 [c]	+	+	+	+

[a] 可基于划分输入输出来识别等价类，为每个等价类选择一个有代表性的测试值。

[b] 该方法用于接口、接近边界的值、与边界交叉的值及超出范围的值。

[c] 错误猜测测试可基于经验学习和专家判断中收集的数据。

9.4.5 为了评估测试案例的完整性并证明没有非预期的功能，应确定在软件单元层面的要求覆盖率，同时应按照表 12 列出的度量对结构覆盖率进行测定。如果认为已实现的结构覆盖率不充分，应定义额外的测试案例或提供接受理由。

示例 1：结构覆盖率分析可以显示基于要求的测试案例的不足、要求的缺陷、无作用码、无效代码或非预期的功能。

示例 2：基于可接受的无作用码（例如：用于调试的代码）或不同软件配置的代码区段可以给出接受所达到的覆盖率水平的理由；或可以使用补充方法（例如：检查）验证未被覆盖的代码。

表 12 软件单元层面的结构覆盖率度量

方法		ASIL			
		A	B	C	D
1a	语句覆盖率	++	++	+	+
1b	分支覆盖率	+	++	++	++
1c	MC/DC（修改条件/决策覆盖率）	+	+	+	++

注 1：通过使用适当的软件工具可以确定结构覆盖率。

注 2：在基于模型开发的情况下，结构覆盖率分析可以利用相似的模型结构覆盖率度量在模型层面进行。

注 3：如果“检测代码”用于确定覆盖率水平，证明检测对测试结果没有影响可能是必要的，这可以通过使用“非检测代码”重复测试实现。

9.4.6 对于软件单元测试的测试环境，应该尽可能地接近目标环境。如果软件单元测试不是在目标环境下执行，应分析源代码和目标代码的差异及测试环境和目标环境之间的差异，以便在后续测试阶段的目标环境中，定义额外的测试。

注 1：测试环境和目标环境之间的差异，可呈现在源代码或目标代码中，例如，由于处理器的数据字和地址字比特宽度的差异。

注 2：根据测试范围，使用适当的测试环境（例如目标处理器、处理器仿真器或开发系统）执行软件单元测试。

注 3：软件单元测试可以在不同的环境中执行，例如：

— 模型在环测试；

— 软件在环测试；

— 处理器在环测试；和

— 硬件在环测试。

注 4：对基于模型的开发，在模型层面执行软件单元测试，随后，在模型和目标代码之间进行背靠背的比较测试。背靠背的比较测试用于确保关于测试对象的模型表现等同于自动生成的代码。

9.5 工作成果

9.5.1 **软件验证计划（细化的）**，由 9.4.2 到 9.4.6 的要求得出。

9.5.2 **软件验证规范**，由 9.4.2 和 9.4.4 至 9.4.6 的要求得出。

9.5.3 **软件验证报告（细化的）**，由 9.4.2 的要求得出。

10 软件集成和测试

10.1 目的

此子阶段的第一个目的是集成软件要素。

此子阶段的第二个目的是证明软件架构已被嵌入式软件实现

10.2 总则

在此子阶段，按照软件架构设计，对软件要素之间特有的集成层次和接口进行测试。软件要素的集成和测试的步骤直接对应着软件的分层架构。

嵌入式软件由安全相关和安全无关的软件要素组成。

10.3 本章的输入

10.3.1 前提条件

应具备以下信息：

— 软硬件接口规范（细化的），按照 6.5.2；

— 软件架构设计规范， 按照 7.5.1；

— 安全计划（细化的）， 按照 7.5.2；

— 软件单元实现， 按照 8.5.2；

— 软件验证计划（细化的），按照 9.5.1

— 软件验证规范，按照 9.5.2；和

— 软件验证报告（细化的），按照 9.5.3。

10.3.2 支持信息

可考虑如下信息：

— 具备经鉴定的软件组件，参见 GB/T XXXXX-8 中第 12 章；

— 软件工具资质报告，按照 GB/T XXXXX-8 中 11.5.2;

— 工具应用指南，按照 5.5.4; 和

— 方法应用指南（来自外部资源）。

10.4 要求和建议

10.4.1 软件集成的计划应描述将各个软件单元分层集成到软件组件中的步骤，直到整个嵌入式软件全部被集成 ，并应考虑：

a) 与软件集成相关的功能依存关系；和
b) 软件集成和软硬件集成之间的依存关系。

注：对基于模型的开发，软件集成可被模型层面的集成和后续由集成的模型生成自动代码来代替。

10.4.2 应按照 GB/T XXXXX-8 第 9 章来计划、定义并执行软件集成测试。

注 1：基于 GB/T XXXXX-8, 第 9 章中的定义， 软件集成测试的对象是软件组件。

注 2：对基于模型的开发，测试对象可以是与软件组件相关的模型。

10.4.3 应使用表 13 列举的软件集成测试方法以证明软件组件和嵌入式软件均实现：

a) 与软件架构设计的符合性，按照第 7 章；
b) 与软硬件接口规范的符合性，按照 GB/T XXXXX-4, 第 7 章；
c) 已定义的功能性；
d) 鲁棒性；

示例：不存在无法访问的软件；具备有效的错误检测和处理。

e) 支持功能的足够资源。

表 13 软件集成测试方法

方法		ASIL			
		A	B	C	D
1a	基于要求的测试 [a]	++	++	++	++
1b	接口测试	++	++	++	++
1c	故障注入测试 [b]	+	+	++	++
1d	资源使用测试 [cd]	+	+	+	++
1e	模型和代码之间的背靠背比较测试，如果适用	+	+	++	++

[a] 架构层面的软件要求是基于要求测试的基础。

[b] 这包括注入任意故障以测试安全机制（例如， 通过破坏软件和硬件组件）

[c] 为了确保受具有足够误差的硬件架构设计影响的要求被完全满足，特性比如平均和最大的处理器性能、最小和最大执行时间、存储使用情况（例如，堆栈使用的 RAM, 程序和数据使用 ROM）以及通信链路的带宽（例如，数据总线）,必须得到确定

[d] 只有在目标硬件上执行软件集成测试或目标处理器的仿真器支持资源测试时，才能正确评估资源使用测试的某些方面。

[e] 这个方法需要一个可以模拟软件组件功能的模型。在这里，通过相同的方式激励模型和代码，并比较彼此输出的结果。

10.4.4 为了能够给软件集成测试方法(按照 10.4.3 选择的)定义恰当的测试案例，应使用表 14 所列的方法。

表 14 软件集成测试案例的得出方法

方法		ASIL			
		A	B	C	D
1a	要求分析	++	++	++	++
1b	等价类生成与分析[a]	+	++	++	++
1c	边界值分析[b]	+	++	++	++
1d	错误推测[c]	+	+	+	+

[a] 可基于划分输入输出来识别等价类，为每个等价类选择一个有代表性的测试值。

[b] 该方法用于接口、接近边界的值、与边界交叉的值及超出范围的值。

[c] 错误猜测测试可基于经验学习和专家判断中收集的数据。

10.4.5 为了评估测试的完整性并确信没有非预期的功能，应确定测试案例在软件架构层面对要求的覆盖率。如果必要，应指定附加的测试案例或者提供理由。

10.4.6 按照 4.3，该子章节适用于 ASIL (A)、(B)、C 和 D 等级：为了评估测试案例的完整性并确信没有非预期的功能，应按照表 15 列出的度量对结构覆盖率进行测定。如果认为结构覆盖率不充分，应定义额外的测试案例或提供可接受的理由。

示例：结构覆盖率分析可以显示基于要求的测试案例的不足、要求的缺陷、无作用码、无效代码或非预期的功能。

表 15 软件架构层的结构覆盖率度量

方法		ASIL			
		A	B	C	D
1a	函数覆盖率[a]	+	+	++	++
1b	调用覆盖率[b]	+	+	++	++

[a] 方法 1a 是指执行的软件函数的百分比。此证据可通过合适的软件集成策略来达成。

[b] 方法 1b 是指执行的软件函数调用的百分比。

注 1：通过适当的软件工具可确定结构覆盖率。

注 2：在基于模型的开发中，使用对模型的类似的结构覆盖度量可在模型级别执行软件架构测试。

10.4.7 应验证作为生产发布（按照 GB/T XXXXX-4 第 11 章）一部分的嵌入式软件，包含了所有已指定的函数，并只包含不损害软件安全要求符合性的其它未指定的函数。

示例：未指定的函数包括用于调试或检测的代码。

注：如果可确保这些未指定的函数不被执行，这是一种符合要求的可接受的方法。否则这些代码的移除是一个变更（参见 GB/T XXXXX-8，第 8 章）。

10.4.8 软件集成测试的测试环境应尽可能接近目标环境。如果集成测试没有在目标环境中执行，应分析源代码和目标代码之间的差异以及测试环境和目标环境之间的差异，来指定后续测试阶段中在目标环境中的附加测试。

注 1：测试环境与目标环境之间的差异可出现在源代码和目标代码中，例如，由于不同处理器的数据字和地址字的不同位宽引起的差异。

注 2：根据测试范围和集成的层级，使用适当的测试环境进行软件要素测试。这些测试环境可以是用于最终集成的目标处理器，或者是用于之前集成步骤的处理器模拟器或开发系统。

注 3：软件集成测试可在不同环境中执行，例如：

— 模型在环测试

— 软件在环测试

— 处理器在环测试；

— 硬件在环测试。

10.5 工作成果

10.5.1 **软件验证计划（细化的）**，由 10.4.1 到 10.4.6 和 10.4.8 的要求得出。

10.5.2 **软件验证规范（细化的）**，由 10.4.1,10.4.2，10.4.4, 10.4.5, 10.4.7 和 10.4.8 的要求得出。

10.5.3 **嵌入式软件**，由 10.4.1 的要求得出。

10.5.4 **软件验证报告（细化的）**，由 10.4.2 的要求得出。

11 软件安全要求验证

11.1 目的

该子阶段的目的是证明嵌入式软件满足软件安全要求。

11.2 总则

软件安全要求验证的目的是证明嵌入式软件在目标环境下满足软件安全要求。

11.3 本章的输入

11.3.1 前提条件

应具备下列信息：

— 软件架构设计规范，按照 7.5.1；

— 安全计划（细化的），按照 7.5.2；

— 软件安全要求规范（细化的），按照 7.5.3；

— 软件验证计划（细化的），按照 10.5.1；

— 软件验证规范（细化的），按照 10.5.2；

— 软件验证报告（细化的），按照 10.5.4；以及

— 集成测试报告，按照 GB/TXXXX-4.8.5.3。

11.3.2 支持信息

可考虑下列信息：

— 验证计划（细化的）（参见 GB/TXXXX-4，6.5.3）；

— 技术安全概念（参见 GB/TXXXX-4，7.5.1）；

— 系统设计规范（参见 GB/TXXXX-4，7.5.2）；

— 工具应用指南，依据 5.5.4；以及

— 方法应用指南（来自外部）。

11.4 要求和建议

11.4.1 应按照 GB/T XXXXX-8 第 9 章，计划、定义和执行软件安全要求的验证。

11.4.2 为验证嵌入式软件是否满足软件安全要求，应在表 16 所列的测试环境中执行测试。

注：已有的测试案例，如来自软件集成测试的测试案例，可以复用。

表16 用于执行软件安全要求验证的测试环境

方法		ASIL			
		A	B	C	D
1a	硬件在环	+	+	++	++
1b	电控单元网络环境[a]	++	++	++	++
1c	整车环境	++	++	++	++
[a] 实例包括集成了部分或全部车辆电气系统的测试台架、试验样车或骡子车，以及部分总线仿真。					

11.4.3 应在目标硬件上执行对软件安全要求实现的测试。

11.4.4 软件安全要求验证结果的评估应考虑：

a) 与期望结果的一致性；

b) 软件安全要求的覆盖率，以及

c) 通过或不通过的准则。

11.5 工作成果

11.5.1 软件验证计划（细化的），由 11.4.1 到 11.4.3 的要求得出。

11.5.2 软件验证规范（细化的）由 11.4.1 到 11.4.3 的要求得出。

11.5.3 软件验证报告（细化的）由 11.4.1 到 11.4.3 的要求得出。

附录 A

（资料性附录）

产品开发软件层面管理的概览和工作流程

表 A.1 提供了产品开发软件层面特定阶段的目的、前提条件和工作成果的概览。

表 A.1 — 产品开发软件层面概览

章	目的	前提条件	工作成果
5 启动产品软件层面的开发	计划和启动软件开发过程中各子阶段的功能安全活动。	项目计划（细化的） （参见GB/T XXXXX-4中5.5.1）； 安全计划 （参见GB/T XXXXX-4中5.5.2）； 技术安全概念 （参见GB/T XXXXX-4中7.5.1）； 系统设计规范 （参见GB/T XXXXX-4中7.5.2）； 相关项集成和测试计划（细化的） （参见 GB/T XXXXX-4 中 7.5.4）。	5.5.1 安全计划（细化的） 5.5.2 软件验证计划 5.5.3 建模和编程语言的设计和编码指南 5.5.4 工具的应用指南
6 软件安全要求的定义	定义软件安全要求。软件安全要求来源于技术安全概念和系统设计规范。 细化软硬件接口要求。 验证软件安全要求和软硬件接口安全要求与技术安全概念和系统设计规范相一致。	技术安全概念 （参见GB/T XXXXX-4中7.5.1）； 系统设计规范 （参见GB/T XXXXX-4中7.5.2）； 软硬件接口规范 （参见GB/T XXXXX-4中7.5.6）； 安全计划 （细化的） （参见5.5.1）； 软件验证计划 （参见5.5.2）	6.5.1软件安全要求规范 6.5.2软硬件接口规范（细化的） 6.5.3软件验证计划（细化的） 6.5.4软件验证报告

7 软件架构设计	开发用于实现软件安全要求的软件架构设计 验证软件架构设计。	安全计划（细化的） （参见 5.5.1）； 建模和编程语言的设计和编码指南（参见 5.5.3）； 软硬件接口规范 （参见 GB/T XXXXX-4 中 7.5.6）； 软件安全要求规范 （参见 6.5.1）； 软件验证计划（细化的） （参见 6.5.3） 软件验证报告（细化的） （参见 6.5.4）	7.5.1 软件架构设计规范 7.5.2 安全计划（细化的） 7.5.3 软件安全要求规范（细化的） 7.5.4 安全分析报告 7.5.5 关联失效分析报告 7.5.6 软件验证报告（细化的）
8 软件单元的设计和实现	按照软件架构设计和相关软件安全要求定义软件单元。 按照定义实现软件单元。 静态验证软件单元的设计和实现。	建模和编程语言的设计及编码指南 （参见 5.5.3）； 软件验证计划（细化的） （参见 6.5.3）； 软件架构设计规范 （参见 7.5.1） 安全计划（细化的） （参见 7.5.2） 软件安全需求规范（细化的） （参见 7.5.3） 软件验证报告（细化的） （参见 7.5.6）	8.5.1 软件单元设计规范 8.5.2 软件单元的实现 8.5.3 软件验证报告（细化的）
9 软件单元测试	证明软件单元满足软件单元设计规范并且不包含非期望的功能。	软硬件接口规范（细化的） （参见 6.5.2） 软件验证计划（细化的） （参见 6.5.3） 安全计划（细化的） （参见 7.5.2） 软件单元设计规范 （参见 8.5.1） 软件单元的实现 （参见 8.5.2） 软件验证报告（细化的） （参见 8.5.3）	9.5.1 软件验证计划（细化的） 9.5.2 软件验证规范 9.5.3 软件验证报告（细化）

10 软件集成和测试	集成软件要素。 证明软件架构设计由嵌入式软件实现。	软硬件接口规范（细化的） （参见 6.5.2） 软件架构设计规范（参见 7.5.1） 安全计划（细化的）（参见 7.5.2） 软件单元实现（参见 8.5.2） 软件验证计划（细化的） （参见 9.5.1） 软件验证规范（细化的） （参见 9.5.2） 软件验证报告（细化的） （参见 9.5.3）	10.5.1 软件验证计划（细化的） 10.5.2 软件验证规范（细化的） 10.5.3 嵌入式软件 10.5.4 软件验证报告（细化的）
11 软件安全要求的验证	证明嵌入式软件满足了软件安全要求。	软件架构设计规范（参见 7.5.1） 安全计划（细化的）（参见 7.5.2） 软件安全要求规范（细化的）（参见 7.5.3） 软件验证计划（细化的）（参见 10.5.1） 软件验证规范（参见 10.5.2） 软件验证报告（细化的） （参见 10.5.4） 集成测试报告（参见 GB/T XXXXX-4 中 8.5.2）；	11.5.1 软件验证计划（细化的） 11.5.2 软件验证规范（细化的） 11.5.3 软件验证报告（细化的）
附录 C 软件配置	使能软件行为的可控变化，以进行不同的应用 使软件行为的可控变化成为可能，以进行不同的应用	参见安全生命周期中与软件配置相关阶段的适用的应用前提条件	C.5.1 配置数据规范 C.5.2 标定数据规范 C.5.3 安全计划（细化的） C.5.4 配置数据 C.5.5 标定数据 C.5.6 软件验证计划（细化的） C.5.7 验证规范 C.5.8 验证报告

附录 B

（资料性附录）

基于模型的开发

B.1 目的

本附录描述了基于模型的车载软件开发概念，概述了其对产品开发软件层面的影响。

B.2 总则

数学建模，已被广泛的应用于许多工程领域，也正逐步在嵌入式软件开发中得到普遍使用。在汽车领域，建模正用于待实现功能（开、闭环控制，监控）的概念捕获中，也用于真实物理系统行为（车辆环境）的仿真中。

建模通常使用商业现成的建模和仿真软件工具。这些软件工具通过半正式的图形化模型来支持系统、软件要素以及它们之间的连接和接口的开发与定义。这些模型采用可编辑的、层级化框图（例如，控制图）以及扩展的状态转换框图（例如状态图）。软件工具提供必要的描述方法、运算技术和解释器/编译器。图形编辑器使复杂模型的直观开发和描述成为可能。使用层级结构的模块化来控制复杂度。模型由明确定义了输入与输出的功能块组成。这些功能块在框图中通过它们接口之间的定向边连接起来，这些定向连接描述了信号流。由此，它们表征了数学模型中的方程式，即将不同要素的接口变量联系起来。连接线代表了有因果动机的行动方向，定义了一个模块的输出是另一个的输入。其它工具特有的建模语言也可用来理出执行顺序和定时。要素的层级可以包含数个改进级别。

这些模型可以被仿真，例如执行。在仿真过程中，计算遵循已定义的执行方向，直到整个模型都被处理完。有许多不同的求解器可用于求解由模型描述的方程。变步长求解器主要用于对车辆和环境建模。对于嵌入式软件的开发，使用的是定步长求解器，它代表了高效代码生成的一个必要的先决条件。

所描述的建模风格在基于模型开发的嵌入式车载软件中得到了广泛应用。典型的，在开发周期的早期，创建控制软件（例如功能模型）的可执行模型和周边系统的模型（例如整车模型）及环境的模型（例如环境模型），并一起进行仿真。这样，能在可接受的计算速度下对高细节度、高复杂性的汽车系统进行建模，并对其行为进行近乎实际的仿真。当整车/环境模型在开发过程中逐步被真实系统及其真实环境所替代，通过代码生成，功能模型可作为在控制单元上实现嵌入式软件的蓝本。

基于模型开发模式的一个特征在于功能模型不仅详述了所想要的功能，而且还提供了设计信息，并最终作为代码生成方法的实施基础。换言之，这样的一个功能模型表征了规范方面以及设计和实现方面。实际上，这些不同的方面反映了功能模型从早期规范模型经过设计模型直到实施模型并最后自动转换成代码（模型的演变）的演变。与具备清晰阶段分隔的基于代码的软件开发相比，在基于模型开发中，可注意到在“软件安全需求”，“软件架构设计”和“软件单元设计和实现”之间更强的阶段合并。此外，在连续的开发阶段中，可使用一种一致的图形建模标记法。验证活动也会变得不同，因为模型可以作为测试过程（例如基于模型的测试）的有用信息来源，或者可以作为对象来验证。无缝地使用模型能够促进高度一致、有效的开发。

注：基于模型开发的样式不局限于上述所提到的已有模型类型。可使用例如 UML 的其它模型。

附录 C
（规范性附录）
软件配置

C.1 目的

软件配置的目的是允许软件行为发生可控的变化，以进行不同的应用。

C.2 总述

可配置软件允许使用配置数据和标定数据开发特定应用的软件（参见图 C.1）。

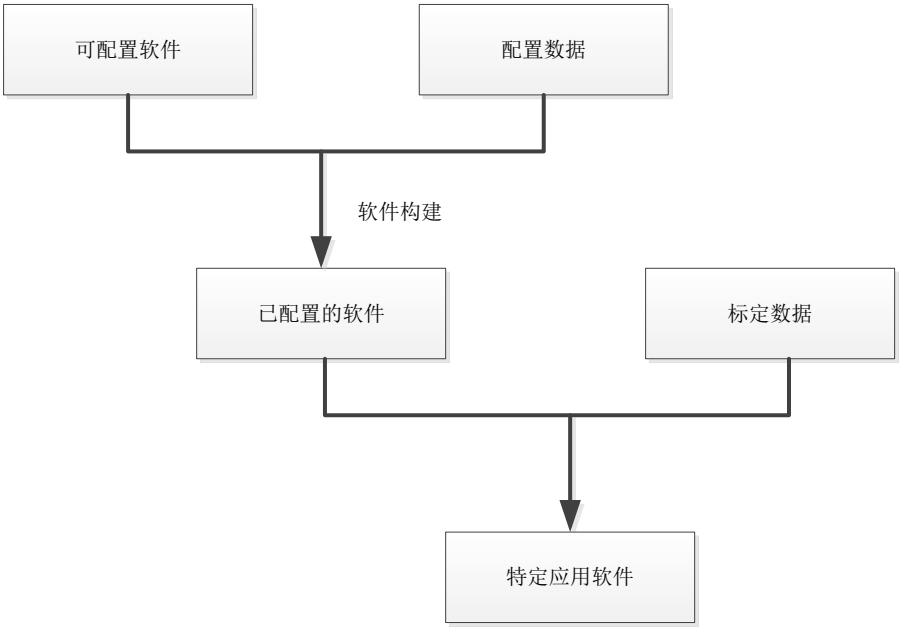


图 C.1 生成特定应用软件

C.3 本章输入

C.3.1 前提条件

前提条件依照应用软件配置的相关阶段。

C.3.2 支持信息

见应用软件配置的相关阶段中的适用的支持信息。

C.4 要求和建议

C.4.1 为确保在安全生命周期中可配置软件的正确使用，应对配置数据进行定义，包含以下内容：

a) 配置数据的有效值；

b) 配置数据的目的和用法；

c) 范围、比例、单位；及

d) 配置数据不同要素之间的相互依赖。

C.4.2 应验证配置数据以确保：

a) 使用的值在其规定的范围内；和

b） 与其他配置数据的兼容性。

注：在软件生命周期的测试阶段（参见第 9 章（软件单元测试）、第 10 章（软件集成和测试）、第 11 章（软件安全要求验证）和 GB/TXXXX -4 第 8 章（相关项集成和测试））测试已配置的软件。

C.4.3 配置数据的 ASIL 等级应等于使用该数据的可配置软件的最高 ASIL 等级。

C.4.4 应根据 GB/T XXXXX-8 第 9 章计划、定义、执行可配置软件的验证。可配置软件应结合所考虑的相关项开发中使用的配置数据组进行验证。

注：只有其行为依赖于配置数据的那部分嵌入式软件要针对配置数据组进行验证。

C.4.5 对于可配置软件，宜应用图 C.2 或 C.3 所示的简化的软件安全生命周期。

注：下列验证活动的组合可实现对已配置软件的完整验证：

a） 可配置软件的验证；

b） 配置数据的验证；及

c） 已配置软件的验证。

这是通过下面任一个来实现

— 在 a）中验证允许的配置数据的范围，并说明它符合在 b）中的范围，或

— 在 b)中说明容许的配置数据范围的符合性并执行 c)。

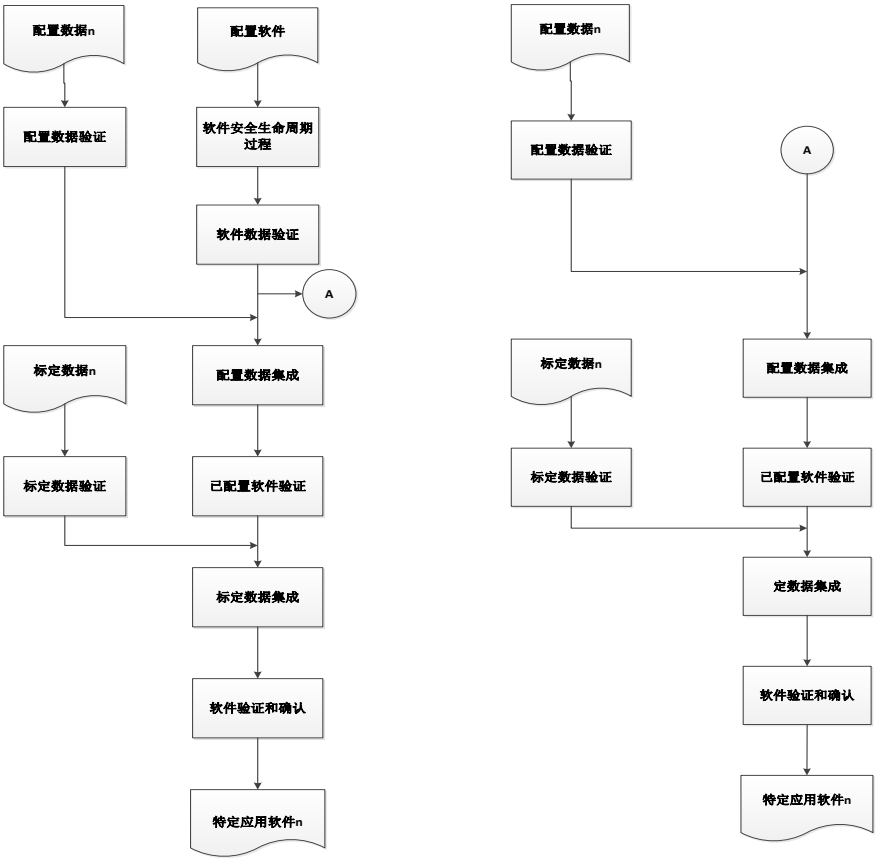


图 C.2—有配置软件和不同配置数据的软件开发的参考模型的变型

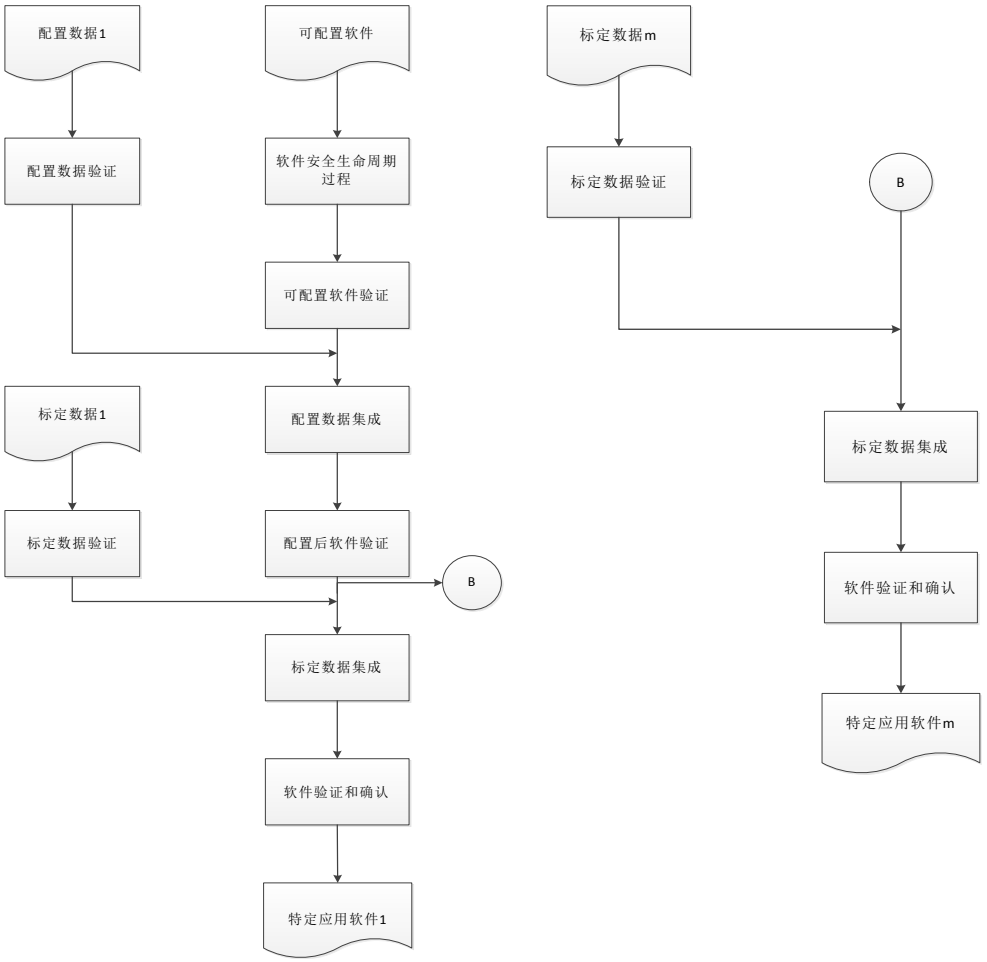


图 C.3-有配置软件和不同标定数据的软件开发的参考模型的变型

C.4.6 应定义与软件组件关联的标定数据以确保配置后软件的正确运行和预期性能，包含

a) 标定数据的有效值；
b) 标定数据的目的和用法；
c) 范围、比例和单位，以及他们对运行状态的依赖（如适用）；
d) 不同标定数据之间的已知的相互依赖；和

注：相互依赖可存在于同一标定数据组内的标定数据之间，或者不同标定数据组的标定数据之间，如：不同电子控制单元的软件中执行的应用于相关功能的标定数据。

e) 配置数据和标定数据之间的已知的相互依赖。

注：配置数据可对使用标定数据的已配置软件有影响。

C4.7 应根据 GB/T XXXXX-8 第 9 章计划、定义、执行标定数据的验证。标定数据的验证应检查标定数据是否在其规定的边界范围内。

注：标定数据的验证，可在特定应用的软件验证时进行，或者在运行时由可配置软件验证；

C4.8 标定数据的 ASIL 等级应等于其可能违反的软件安全需求的最高 ASIL 等级。

C4.9 为检测安全相关的标定数据的非预期变更，应使用表 C.1 中所列的数据非预期变化的检测机制。

表 C.1-数据非预期变化的检测机制

方法		ASIL			
		A	B	C	D
1a	标定数据真实性检查	++	++	++	++
1b	标定数据的冗余存储	+	+	+	++
1c	错误检测码 [a]	+	+	+	++
[a] 错误检测码也可按照 GB/T XXXXX-5 在硬件中实现					

C.4.10 计划标定数据的生成和应用时，应定义：

a) 应遵循的流程；
b) 生成标定数据的工具,以及
c) 验证标定数据的流程。

注：标定数据的验证可包含检查标定数据值的范围或者不同标定数据之间的相互依赖关系。

C.5 工作成果

C5.1 配置数据规范，由 C4.1 和 C4.3 的要求得出。

C5.2 标定数据规范，由 C4.6 的要求得出。

C5.3 安全计划（细化），由 C.4.1, C.4.4, C.4.5, C.4.9 和 C.4.10 要求得出。

C5.4 配置数据，由 C.4.3 的要求得出。

C5.5 标定数据，由 C.4.8 的要求得出。

C5.6 软件验证计划（细化），由 C.4.2, C.4.4, C.4.7 和 C.4.10 的要求得出。

C.5.7 验证规范，由 C.4.4 和 C.4.7 的要求得出。

C.5.8 验证报告，由 C.4.1, C.4.4, C.4.7 和 C.4.8.的要求得出。

附录 D

（资料性附录）

避免软件要素间的相互干扰

D.1 目的

本附录的目的是提供能引起软件要素（例：不同软件划分中的软件要素）相互干扰的故障示例，此外本附录也提供一些可用于预防、检测和减轻所列出的故障的处理机制的示例。

注：在开发过程中，对预防、检测和减轻相关故障的处理机制的能力和有效性进行评估。

D.2 总则

D.2.1 免于干扰的达成

为了开发软件要素间的不相互干扰或评估其达成情况，要考虑典型故障的影响和可能的由此产生的失效的蔓延。

D.2.2 时序和执行

关于时间限制，每个软件分割中执行的软件要素需考虑如下所罗列的故障影响：

— 执行阻塞;

— 死锁;

— 活锁；

— 执行时间的不正确分配;

— 软件要素间的不正确同步。

示例：需要考虑的处理机制如：循环执行调度、固定优先级调度、时间触发调度、处理器执行时间监控、程序执行次序监控和到达（发生）率监控。

D.2.3 内存

关于内存，每个软件分割中执行的软件要素需考虑如下所罗列的故障影响：

— 内容损坏；

— 对已分配给其它软件要素的内存进行读写访问。

示例：可使用的处理机制如：内存保护、奇偶校验位、纠错码（ECC）、循环冗余校验（CRC）、冗余存储、内存访问限制、内存访问的静态分析、内存静态分配。

D.2.4 信息交换

关于信息交换，针对每个发送方或接收方，需要考虑如以下所列的那些故障的原因或故障的影响：：

— 信息重复；

— 信息丢失；

— 信息延迟；

— 信息插入；

— 信息伪装或信息的不正确寻址；

— 信息次序不正确；

— 信息损坏；

— 从发送方传送到多个接收者的信息不对称；

— 发送方发送的信息只能被部分接收方接收；

— 通信信道阻塞。

注：在不同软件分割或 ECU 中的要素间的信息交换包括信号、数据、消息等。

示例 1：可使用 I/O 设备、数据总线等方式进行信息交换。

示例 2：可使用的处理机制如：通信协议、信息重复、信息回送、信息确认、I/O 引脚的适当配置、分离的点到点的单向通信对象、明确的双向通信对象、异步数据通信、同步数据通信、事件触发数据总线、带有时间触发访问的事件触发数据总线、时间触发的数据总线、最小时间片和基于优先级的总线仲裁。

示例 3：通信协议包含的信息有通信对象的标识符、保持活动的消息、激活的计数器、序列号、错误检测码和纠错码。