

引言

本文档描述如何在安全相关系统的背景下使用STM32L0系列微控制器，并指定了为达到目标安全完整性等级，用户需承担的安装和操作责任。

本手册适用于属于STM32L0系列的微控制器和X-CUBE-STL软件包。

如果遵循本手册的指示，则系统设计者无需了解STM32L0系列功能安全标准应用的详情。

本手册按照IEC 61508标准编写。它描述了如何在其他功能安全标准（例如，机械安全指示ISO 13849）的背景下使用STM32L0系列微控制器。

本手册中收集的安全分析考虑了基于Arm[®] Cortex[®]-M3的STM32L0xx微控制器的不同产品型号中存储器大小、内部外设可用性和计数以及封装的变化。

本手册必须与相关产品型号的技术文档（例如参考手册和数据手册）一起阅读，这些技术文档可以在www.st.com上获取。

目录

1	关于本文档	9
1.1	目的和范围	9
1.2	术语和缩略语	9
1.3	参考标准	10
2	STM32L0系列产品开发过程	12
2.1	STMicroelectronics标准开发过程	12
3	参考安全架构	14
3.1	引言	14
3.2	合规项	14
3.2.1	合规项的定义	14
3.2.2	合规项执行的安全功能	15
3.2.3	参考安全架构 - 1001	15
3.2.4	参考安全架构 - 1002	16
3.3	假定要求	17
3.3.1	假定安全要求	17
3.4	电气规范和环境限制	19
3.5	系统安全完整性	19
3.6	硬件和软件诊断说明	20
3.6.1	Arm® Cortex®-M3 CPU	21
3.6.2	嵌入式FLASH	26
3.6.3	内部 SRAM	29
3.6.4	系统总线架构	32
3.6.5	NVIC和EXTI控制器	34
3.6.6	直接存储器访问控制器 (DMA)	35
3.6.7	通用同步收发器 (USART1/2/4/5)、低功耗通用异步收发器 (LPUART1)	38
3.6.8	内部集成电路 (I2C1/2)	40
3.6.9	串行外设接口 (SPI1/2)	42
3.6.10	USB - 2.0通用串行总线接口FS模块	44
3.6.11	触摸感应控制器 (TSC)	46
3.6.12	模数转换器 (ADC)	48
3.6.13	数模转换器 (DAC)	50

3.6.14	比较器 (COMP)	51
3.6.15	基本定时器 (TIM 6/7)	53
3.6.16	高级、通用和低功耗定时器 (TIM/2/3/21/22, LPTIM1/2)	54
3.6.17	通用输入/输出 (GPIO) - 端口A/B/C/D/E/F/G/H	57
3.6.18	实时时钟模块 (RTC)	59
3.6.19	复位和时钟控制 (RCC) 子系统	64
3.6.20	独立看门狗 (IWDG), 系统窗口看门狗 (WWDG)	66
3.6.21	调试	67
3.6.22	循环冗余校验模块 (CRC)	67
3.6.23	系统配置控制器 (SYSCFG)	68
3.6.24	灵活的静态存储控制器 (FSMC)	69
3.6.25	真随机数发生器 (RNG)	71
3.6.26	高级加密标准硬件加速器 (AES)	72
3.6.27	防火墙 (FW)	73
3.6.28	液晶显示控制器 (LCD)	74
3.6.29	禁用并定期交叉检查未使用外设的意外激活	75
3.7	使用条件	77
4	安全结果	82
4.1	随机硬件故障安全结果	82
4.1.1	安全分析结果自定义	83
4.1.2	免受干扰 (FFI) 的一般要求	83
4.1.3	关于多故障情景的说明	84
4.2	从属故障分析	84
4.2.1	电源	84
4.2.2	时钟	84
4.2.3	DMA	85
4.2.4	内部温度	85
5	证据列表	86
附录A	其他安全标准的变更影响分析	87
A.1	ISO 13849-1 / ISO 13849-2	87
A.1.1	架构类别	87
A.1.2	安全指标计算	89
A.1.3	工作成果	90
A.2	IEC 62061:2005/AMD1:2012	92

A.2.1	架构类别	93
A.2.2	安全指标计算	95
A.2.3	工作成果	95
A.3	IEC 61800-5-2:2007	96
A.3.1	架构类别	97
A.3.2	安全指标计算	97
A.3.3	工作成果	97
A.4	ISO 26262:2010	98
A.4.1	架构类别	99
A.4.2	安全指标计算	99
A.4.3	工作成果	100
版本历史		101



表格索引

表1.	术语和缩略语	9
表2.	本文档与IEC 61508-2附录D要求之间的对应关系	11
表3.	SS1和SS2安全状态详细信息	19
表4.	安全机制字段说明	20
表5.	CPU_SM_0	21
表6.	CPU_SM_1	21
表7.	CPU_SM_2	22
表8.	CPU_SM_3	23
表9.	CPU_SM_4	23
表10.	CPU_SM_5	24
表11.	CPU_SM_6	24
表12.	CPU_SM_7	25
表13.	MPU_SM_0	25
表14.	FLASH_SM_0	26
表15.	FLASH_SM_1	26
表16.	FLASH_SM_2	27
表17.	FLASH_SM_3	27
表18.	FLASH_SM_4	28
表19.	FLASH_SM_5	28
表20.	FLASH_SM_6	28
表21.	FLASH_SM_8	29
表22.	RAM_SM_0	29
表23.	RAM_SM_2	30
表24.	RAM_SM_3	30
表25.	RAM_SM_4	31
表26.	RAM_SM_5	32
表27.	BUS_SM_0	32
表28.	BUS_SM_1	33
表29.	LOCK_SM_0	33
表30.	NVIC_SM_0	34
表31.	NVIC_SM_1	34
表32.	DMA_SM_0	35
表33.	DMA_SM_1	36
表34.	DMA_SM_2	36
表35.	DMA_SM_3	37
表36.	DMA_SM_4	37
表37.	UART_SM_0	38
表38.	UART_SM_1	38
表39.	UART_SM_2	39
表40.	UART_SM_3	39
表41.	IIC_SM_0	40
表42.	IIC_SM_1	40
表43.	IIC_SM_2	41
表44.	IIC_SM_3	41
表45.	IIC_SM_4	42
表46.	SPI_SM_0	42
表47.	SPI_SM_1	43
表48.	SPI_SM_2	43

表49.	SPI_SM_3	44
表50.	SPI_SM_4	44
表51.	USB_SM_0	44
表52.	USB_SM_1	45
表53.	USB_SM_2	45
表54.	USB_SM_3	46
表55.	TSC_SM_0	46
表56.	TSC_SM_1	47
表57.	TSC_SM_2	47
表58.	ADC_SM_0	48
表59.	ADC_SM_1	48
表60.	ADC_SM_2	49
表61.	ADC_SM_3	49
表62.	ADC_SM_4	50
表63.	DAC_SM_0	50
表64.	DAC_SM_1	51
表65.	COMP_SM_0	51
表66.	COMP_SM_1	52
表67.	COMP_SM_2	52
表68.	COMP_SM_3	52
表69.	COMP_SM_4	53
表70.	GTIM_SM_0	53
表71.	GTIM_SM_1	54
表72.	ATIM_SM_0	54
表73.	ATIM_SM_1	55
表74.	ATIM_SM_2	56
表75.	ATIM_SM_3	56
表76.	ATIM_SM_4	57
表77.	GPIO_SM_0	57
表78.	GPIO_SM_1	58
表79.	GPIO_SM_2	58
表80.	GPIO_SM_3	59
表81.	RTC_SM_0	59
表82.	RTC_SM_1	60
表83.	RTC_SM_2	60
表84.	RTC_SM_3	61
表85.	VSUP_SM_0	61
表86.	VSUP_SM_1	62
表87.	VSUP_SM_2	62
表88.	VSUP_SM_3	63
表89.	VSUP_SM_4	63
表90.	CLK_SM_0	64
表91.	CLK_SM_1	64
表92.	CLK_SM_2	65
表93.	CLK_SM_3	65
表94.	WDG_SM_0	66
表95.	WDG_SM_1	66
表96.	DBG_SM_0	67
表97.	CRC_SM_0	67
表98.	SYSCFG_SM_0	68
表99.	DIAG_SM_0	68
表100.	FSMC_SM_0	69

表101.	FSMC_SM_1	69
表102.	FSMC_SM_2	70
表103.	FSMC_SM_3	70
表104.	RNG_SM_0	71
表105.	RNG_SM_1	71
表106.	AES_SM_0	72
表107.	AES_SM_1	72
表108.	AES_SM_2	73
表109.	FWR_SM_0	73
表110.	LCD_SM_0	74
表111.	LCD_SM_1	74
表112.	FFI_SM_0	75
表113.	FFI_SM_1	75
表114.	安全机制列表	77
表115.	整体可达到的安全完整性等级	82
表116.	FFI的一般要求列表	83
表117.	ISO 13849架构分类	88
表118.	ISO 13849工作成果列表	90
表119.	SIL分类与HFT	93
表120.	IEC 62061架构分类	94
表121.	IEC 62061工作成果列表	96
表122.	IEC 61800工作成果列表	97
表123.	IEC 26262工作成果列表	100
表124.	文档版本历史	101
表125.	中文文档版本历史	101

图片索引

图1. STMicroelectronics产品开发过程 13

图2. 合规项的定义 14

图3. 1oo1参考架构 16

图4. 1oo2参考架构 17

图5. STM32 PST的分配和目标..... 18

图6. ISO 13849类的框图 B类和1类的框图 1 88

图7. ISO 13849类的框图 2 89

图8. ISO 13849类的框图 3类和4类的框图 4. 89

图9. SRECS框图 95

图10. SIL和ASIL之间的相关模型 99



1 关于本文档

1.1 目的和范围

本文档描述如何在安全相关系统的背景下使用基于Arm^{®(a)}内核的STM32L0xx微控制器，并指定了为达到所需安全完整性等级，用户需承担的安装和操作责任。

对于内置一个或多个STM32L0系列微控制器的解决方案，系统设计者可使用本文档评估该解决方案的安全性。



1.2 术语和缩略语

STM32L0系列硬件模块的相关缩略语（例如，DMA、GPIO等）与STM32L0系列技术文档中使用的相同。参见 [表 1](#) 获取本文档中所用缩略语的列表。

表1. 术语和缩略语

缩略语	定义
CCF	共因故障
CM	连续模式
COTS	商用现成品
CoU	使用条件
CPU	中央处理器
CRC	循环冗余校验
DC	诊断覆盖率
DMA	直接存储器访问
DTI	诊断测试间隔
ECM	发动机控制模块
ECU	电子控制单元
EUC	受控设备
FIT	故障率
FMEA	故障模式影响分析
FMEDA	故障模式影响诊断分析
HD	高需求
HFT	硬件容错
HW	硬件
ITRS	国际半导体技术发展路线图

a. Arm是Arm Limited（或其子公司）在美国和/或其他地区的注册商标。

表1. 术语和缩略语（续）

缩略语	定义
LD	低需求
MCU	微控制器单元
MTBF	平均故障间隔时间
MTTFd	系统平均无危险故障时间
NA	不可用
PDS(SR)	功率驱动系统（安全相关）
PEc	可编程电子设备 - 核心
PEd	可编程电子设备 - 诊断
PFD	按需情况下发生危险故障的概率
PFH	每小时故障概率
PL	性能等级
PST	过程安全时间
SFF	安全失效分数
SIL	安全完整性等级
SRCF	安全相关控制功能
SRECS	安全相关电气控制系统
SRP/CS	控制系统的安全相关部件
SW	软件

另请阅读本手册中使用的以下定义：

- 终端用户：STM32L0系列的最终用户，负责将MCU集成到实际应用（例如，电子控制板）中。
- 应用软件：在STM32L0系列MCU上运行并实现安全功能的实际软件。

1.3 参考标准

本文档按照适用于电气、电子和可编程电子安全相关系统的功能安全的IEC 61508国际标准编写。

所参考的版本为IEC 61508:1-7 © IEC:2010。

本手册中考虑的其他功能安全标准如下：

- ISO 26262-1, 2, 3, 4, 5, 6, 7, 8, 9: 2011(E), ISO 26262-10: 2012(E)
- ISO 13849-1:2006, ISO 13849-2:2010
- IEC 62061:2012-11, 版本 1.1
- IEC 61800-5-2:2007, 版本1.0



表 2 列出了本文档内容与IEC 61508-2附录D中所列要求的对应关系。

表2. 本文档与IEC 61508-2附录D要求之间的对应关系

IEC 61508要求（第2部分附录D）	参考
D2.1 a) 能够执行的功能的功能说明	第 3 节
D2.1 b) 合规项硬件和/或软件配置的识别	第 3.2 节
D2.1 c) 合规项的使用限制或项目行为或故障率分析所基于的假设	第 3.2 节
D2.2 a) 随机硬件故障导致的合规项故障模式，造成功能故障且不会被合规项内部诊断检测到；	第 3.7 节
D2.2 b) a)中所述每一种故障模式的估计故障率；	
D2.2 c) 随机硬件故障导致的合规项故障模式，造成功能故障并被合规项内部诊断检测到；	
D2.2 d) 随机硬件故障导致的合规项内部诊断故障模式，造成检测功能故障的诊断失败；	
D2.2 e) c)和d)中所述每一种故障模式的估计故障率；	第 3.2.2 节
D2.2 f) 合规项内部诊断检测到的c)中的每一种故障模式的诊断测试间隔；	
D2.2 g) c)中每一种故障模式的内部诊断指示的合规项输出；	第 3.6 节
D2.2 h) 任何定期的验证测试和/或维护要求；	第 3.7 节
D2.2 i) 就指定功能而言，对于能够被内部诊断检测到的这些故障模式，必须提供充足的信息以便开发外部诊断能力。	
D2.2 j) 硬件容错；	第 3 节
D2.2 k) 提供功能的合规项该部分的A型或B型分类（参见7.4.4.1.2和7.4.4.1.3）；	

本手册中报告的安全故障分数是在本文档所述假设下并特别根据第 3.7 节：使用条件中所述使用条件计算得出。

2 STM32L0系列产品开发过程

对于严格要求安全性的应用所使用的微电子器件，它们的开发过程进行了充分管理，以降低设计阶段引入系统故障的可能性。

IEC 61508:2附录F（ASIC技术和措施 - 避免系统故障）充当按照IEC 61508的要求定制微控制器标准设计和制造商过程的指导原则。附录F中报告的核查表有助于收集给定实际过程的所有相关证据。

2.1 STMicroelectronics标准开发过程

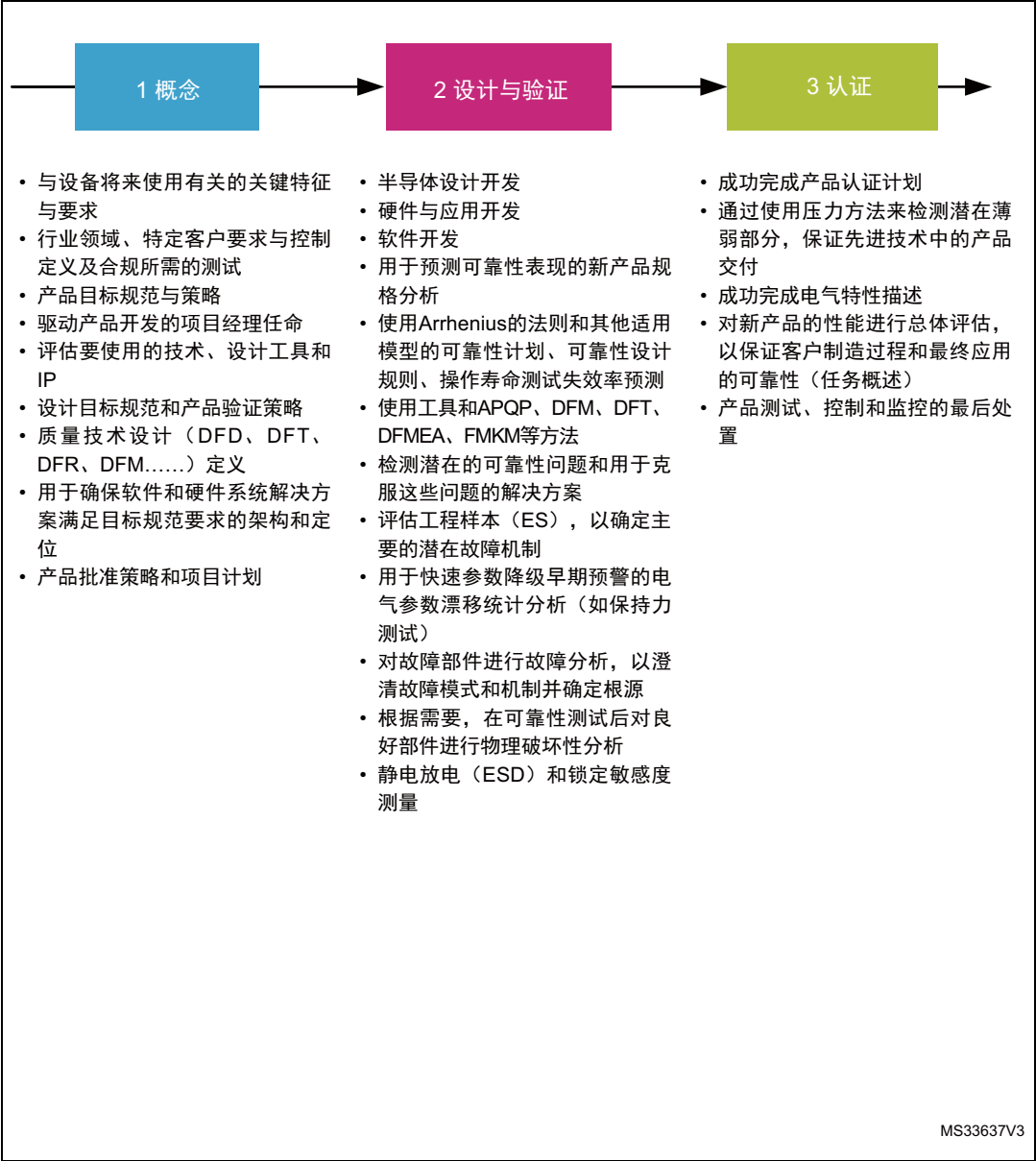
STMicroelectronics（ST）服务于四个工业领域：

- 标准产品。
- 汽车产品：ST汽车产品符合AEC-Q100标准。它们将接受特定的压力测试和处理指令，以达到要求的质量级别和产品稳定性。
- 汽车安全：汽车领域的一个子集。ST以ISO 26262道路车辆功能安全标准为参考。ST支持客户查询产品故障率和FMEDA，为使硬件系统符合既定安全目标提供支持。ST提供可安全应用于预定用途的产品，与客户一起分析任务资料，采用常用方法并为残余风险制定对策。
- 医用品：ST遵守适用的医用品规范，并在这些产品的开发和确认过程中执行尽职调查。

STMicroelectronics产品开发过程符合ISO/TS 16949标准，是一组专用于将客户说明和市场或工业领域要求转化为半导体器件及其所有相关元件（封装、模块、子系统、应用、硬件、软件和文档）的相关活动，符合ST内部程序并能使用ST内部或分包技术进行制造。

 1总结了STMicroelectronics产品开发过程。

图1. STMicroelectronics产品开发过程



3 参考安全架构

3.1 引言

本文档中分析的STM32L0系列微控制器可用作不同安全应用中的合规项。

本节的目的是识别此类合规项，从而根据参考概念定义的相关假设定义分析背景。因此，此概念定义还包含参考安全要求作为已定义合规项之外的设计的假设。

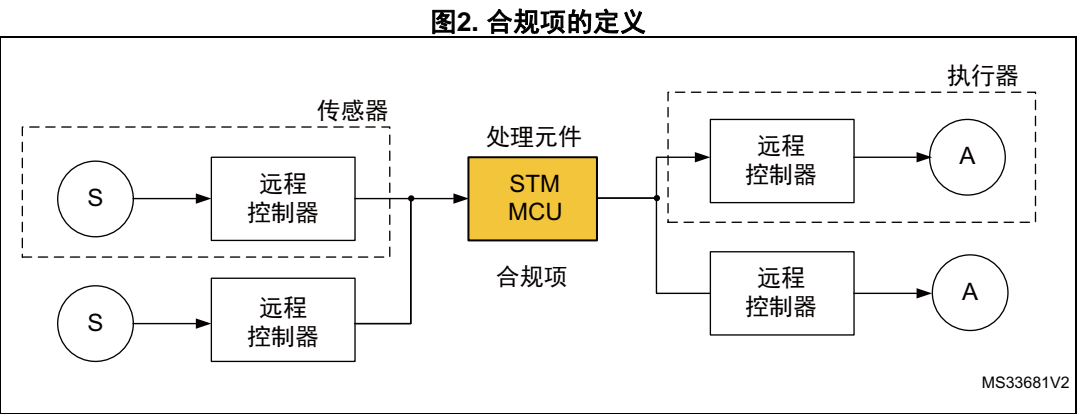
因此，合规项方法的目的是不是提供微控制器所属系统的详尽危险和风险分析，而是列出分析期间考虑的系统相关信息。此类信息包括危险因素的应用相关假设、故障频率和应用已保证的诊断覆盖率等。

3.2 合规项

3.2.1 合规项的定义

根据IEC 61508:1第8.2.12款，合规项是按照IEC 61508系列条款声明的任何项目（例如元件）。在其开发结束时，其用户必须通过安全手册对其进行描述。

在本文档中，合规项被定义为包含一个或两个STM32微控制器的系统（参见图 2）。通信总线直接或间接连接到传感器和执行器。



为保证STM32L0系列的功能（外部存储器、时钟石英等）或其安全性（例如，外部看门狗、电压监控器），需要其他可能与合规项有关的元件（例如，外部硬件元件）。

定义的合规项可按照IEC61508-4第3.4.5款分类为“元件”。

3.2.2 合规项执行的安全功能

合规项架构可以描述为由执行安全功能或部分安全功能的以下过程组成：

- 输入处理元件（PEi）从连接到传感器的远程控制器读取安全相关数据，并将其传输至以下计算元件；
- 计算处理元件（PEc）执行安全功能所需的算法，并将结果传输至以下输出元件；
- 输出处理元件（PEo）将安全相关数据传输至连接到执行器的远程控制器；
- 对于1oo2架构，可能还存在投票处理元件（PEv）；
- 为了保证安全完整性，考虑合规项外部处理，例如看门狗（WDT_e）和电压监控器（VMON_e）。

在详述CoU（安全机制的定义）的章节中阐明了PEv以及外部处理WDT_e和VMON_e的角色：

- WDT_e：参见“独立看门狗”– VSUP_SM_2和“应用软件中的控制流监控”– CPU_SM_1，
- VMON_e：参见“电源电压监控”– VSUP_SM_1。

总之，STM32L0系列微控制器为实现包含以下三项操作的终端用户安全功能提供支持：

- 从输入外设安全采集安全相关数据。
- 应用程序程序的安全执行和相关数据的安全计算。
- 结果或决策到输出外设的安全传输。

使用这三种基础操作完成合规项声明与安全指标计算。

根据上文报告的已实现安全功能的定义，可将该合规项（即元件）视为B类（根据IEC61508-2第7.4.4.1.2款的定义）。尽管对STM32L0系列执行了精确、彻底且详细的故障分析，还必须考虑该器件的内在复杂性，因此分类为B类是合适的。

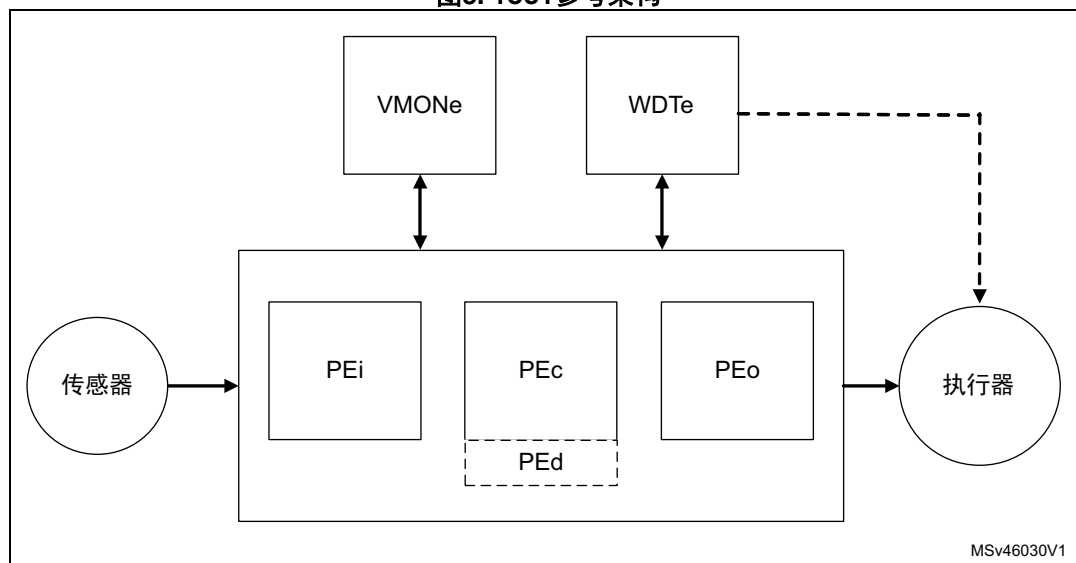
因此，确定了两种主要的安全架构：1oo1（使用一个MCU）和1oo2（使用两个MCU）。

3.2.3 参考安全架构 - 1oo1

在1oo1参考架构（如图 3所示）中，通过STM32L0系列内部处理（已实现安全机制）和外部处理WDT_e与VMON_e的组合来保证合规项的安全完整性。

1oo1参考架构的目标是SIL2。

图3. 1oo1参考架构

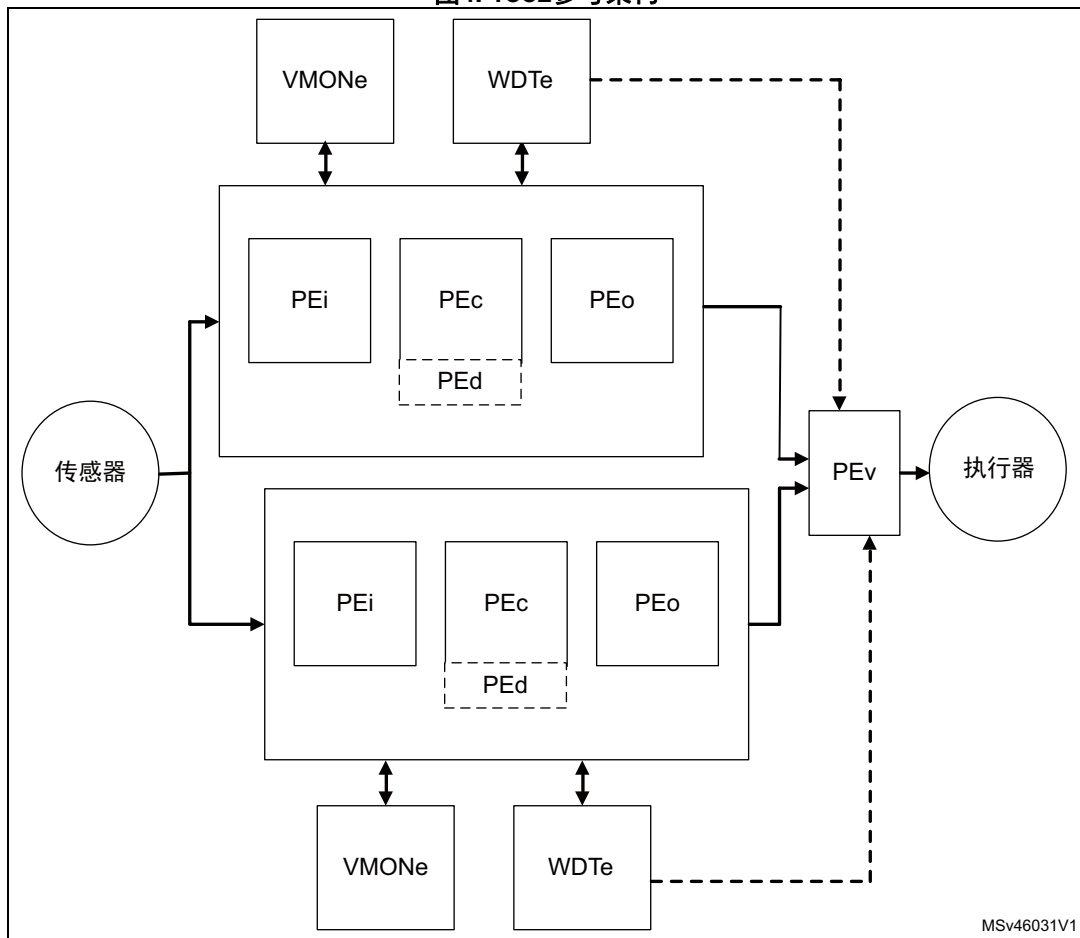


3.2.4 参考安全架构 - 1oo2

1oo2参考架构（如[图 4](#)所示）包含两个独立通道，二者均以与1oo1参考架构相同的方式来实现。通过STM32L0系列内部处理（已实现安全机制）和外部处理WDTe与VMONe的组合来保证每个通道的安全完整性。通过允许声明HFT = 1的外部表决器PEv保证整个合规项的安全完整性。因此，可以达到IEC61508-2表3中规定的更高安全完整性等级。应在两个通道间实现适当隔离（包括电源隔离），以避免共因故障的巨大影响（参见[第 4.2 节](#)）。无论如何，都需要进行 β D计算。

1oo2参考架构的目标是SIL3。

图4. 1oo2参考架构



3.3 假定要求

3.3.1 假定安全要求

概念说明、危险和风险分析、整体安全要求说明和结果分配决定了下列合规项要求（ASR：假定安全要求）。

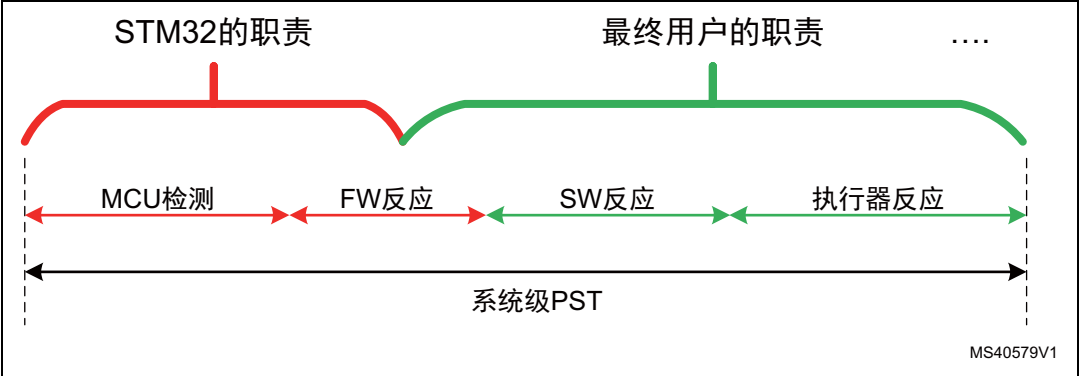
注意： 最终用户负责检查最终应用是否符合这些假设。

ASR1： 根据第4部分第3.5.16款，合规项可用于四种类型的安全功能操作模式：

- 连续模式或高需求SIL3安全功能（CM3），或者
- 低需求SIL3安全功能（LD3），或者
- 连续模式或高需求SIL2安全功能（CM2），或者
- 低需求SIL2安全功能（LD2）。

ASR2： 合规项用于实现允许10 ms（最坏情况）时间预算的安全功能，以便STM32 MCU检测和响应故障。在系统层面的错误响应链中，这段时间相当于分配给STM32L0系列MCU的过程安全时间（[图 5](#)所示“STM32职责”）。

图5. STM32 PST的分配和目标



ASR3: 合规项用在可连续通电超过8小时的安全功能中。假设不需要任何验证测试且产品的生命周期被认为不小于10年。

ASR4: 假定只执行一项安全功能，或者如果很多，则将所有功能划分为相同SIL，从而使它们在安全要求方面不可区分。

ASR5: 如果有多重安全功能实现，则假定终端用户负责保证其需要的相互独立性。

ASR6: 假设应用软件中没有已实现并与安全功能共存的“非安全相关”功能。

ASR7: 假设已实现的安全功能不依赖于STM32L0 MCU至/从低功耗状态的过渡。

ASR8: 合规项的局部安全状态是指以下任何一种状态：

- SS1: 通知应用软件存在故障且可通过应用软件本身响应
- SS2: 无法通知应用软件存在故障或应用软件无法执行响应^(a)

表 3提供了安全状态SS1和SS2的详细信息。

a. 终端用户必须考虑到，影响STM32的随机硬件故障可能危害MCU正常工作的能力（例如，影响程序计数器的故障模式会妨碍软件的正确执行）。

表3. SS1和SS2安全状态详细信息

安全状态	条件	合规项操作	系统过渡到安全状态 – 1oo1架构	系统过渡到安全状态 – 1oo2架构
SS1	通知应用软件存在故障且可通过应用软件本身响应。	向应用软件报告故障	应用软件在安全状态下驱动整个系统	两个通道之一的应用软件在安全状态下驱动整个系统
SS2	无法通知应用软件存在故障或应用软件无法执行响应。	WDTe发出复位信号	WDTe在安全状态 (“安全关闭”) 下驱动整个系统 ⁽¹⁾	PEv在安全状态下驱动整个系统

1. 这里的安全状态达成符合IEC61508-2第7.4.8.1款a)项的相关注释

ASR9: 假定终端用户在系统层面定义的安全状态与合规项的假定局部安全状态 (SS1、SS2) 兼容。

ASR10: 假定按照IEC 61508-2的方法1H和1S分析合规项。^(a)

ASR11: 假设按照IEC61508:2第7.4.4.1.2款将合规项视为B类。

ASR12: 假定在最终系统的维护状态下使用双闪存区擦除和重新编程功能。

3.4 电气规范和环境限制

为保证STM32L0系列的安全完整性，用户不得逾越STM32L0系列用户手册中报告的下列电气规范和环境限制：

- 绝对最大额定值
- 容量
- 工作条件。

由于存在大量STM32L0系列产品编号，本文档中没有列出相关用户手册和数据手册；用户有责任仔细检查相关产品编号的技术文档（参见www.st.com）中的上述限制。

3.5 系统安全完整性

根据IEC 61508-2第7.4.2.2款的要求，STM32L0系列的开发过程中考虑了方法1S。根据IEC61508-2第7.4.6.1款的明确认可，可将STM32 MCU系列视为标准的大量生产的电子集成器件 - 严格的开发程序、严谨的测试和丰富的使用经验最大限度降低了设计故障发生的可能性。总之，已使用IEC 61508-2附录F中建议的技术和措施执行了STM32 MCU开发流程的内部合规评估。安全案例数据库（[第 5 节：证据列表](#)）保存了符合规范的证据。

a. 请参见[第 3.5 节](#)和[第 3.6 节](#)。

3.6 硬件和软件诊断说明

本节列出了STM32L0系列微控制器的安全分析中考虑的所有安全机制（硬件、软件和应用层面）。用户应熟悉STM32L0系列的架构，并将本文档与相关器件的数据手册、用户手册和参考信息结合使用。因此，为避免发生错误并减少要显示的信息量，本文档中只包含了最低限度的功能细节。在下面的描述中，“安全机制”、“方法”或“要求”将作为同义词使用。

请注意，STM32L0系列的各个产品编号拥有不同的外设组合（例如，其中的一些没有配备USB外设）。为了减少文档数量并避免无意义的重复，本手册（以及本节）适用于目标产品编号的所有可用外设。用户必须选择他们的设备上实际提供的外设，并相应地忽略无意义的建议。

本节中提供的实现指导原则仅供参考。STM32L0系列安全分析期间ST执行的安全验证和本手册（或其附录）中记录的相关诊断覆盖率数据都基于此类指导原则。为了清晰起见，针对MCU基础功能对安全机制进行了分组。

以表格的形式组织信息（每种安全机制一个）。表 4解释了每个字段。

表4. 安全机制字段说明

SM代码	唯一的安全机制代码/标识符，也用在FMEA文档中。标识符使用mmm_SM_x模式，其中mmm是3或4个字母的模块缩写，“x”是递增编号。请注意，模块缩写和编号可能不按顺序和/或不同于通过原始文档获得的模块实际名称。
说明	简短的助记说明。
所有权	ST：表示芯片上提供该方法 终端用户：终端用户必须通过应用软件修改、硬件解决方案或这两种方式实现方法。
具体实现	具体实现有时包括引入安全机制后的安全理念说明。
错误报告	描述如何向应用软件报告故障检测。
故障检测时间	安全机制检测硬件故障所需的时间。
已解决故障模型	报告通过诊断解决的故障模式（永久、瞬时或二者兼有）以及其他信息： – 如果归类用于故障避免：有助于降低故障发生概率的方法 – 如果归类用于系统故障：设计用于缓解应用软件设计中的系统错误（问题）的方法。
取决于MCU配置	报告STM32L0系列的不同产品编号的安全机制实现或特性是否有所不同。
初始化	为了激活安全机制的作用要执行的特定操作。
周期性	– 连续：安全机制以连续模式激活 – 定期：定期执行安全机制。请注意，仅当每PST至少执行一次时，安全机制才会影响诊断覆盖率 – 按需：在发生指定事件（例如，数据消息的接收）时激活安全机制 – 启动：应只在上电或离线维护期间执行安全机制。
诊断测试	报告特定程序（如果有并已推荐）以便在线测试安全机制的效力。
多重故障保护	报告相关的安全机制以便正确管理多故障情景（参见第 4.1.3 节：关于多故障情景的说明）。
建议和已知限制	其他字段中没有提供的额外建议或限制（如果有）。



3.6.1 Arm® Cortex®-M3 CPU

表5. CPU_SM_0

SM代码	CPU_SM_0
说明	Arm® Cortex®-M3 CPU的定期核心自测软件
所有权	终端用户或ST
具体实现	软件测试围绕IEC 61508:7第A.3.2款（软件自测：漫步位（单通道））已处理的常见技术进行构建。为了达到要求的覆盖率，通过具体分析所有CPU故障模式和相关故障模式的分布来指定自测软件。
错误报告	取决于具体实现
故障检测时间	取决于具体实现
已解决故障模型	永久
取决于MCU配置	无
初始化	无
周期性	定期
诊断测试	可根据选择的测试实现设计策略，在软件中嵌入自诊断功能。建议采用结果变量校验和保护和防御性编程。
多重故障保护	CPU_SM_5：外部看门狗
建议和已知限制	该方法是STM32L0系列安全理念的主要内容。由于MCU外设的已定义诊断的主要部分以软件为基础，因此CPU完整性是一个关键因素。

表6. CPU_SM_1

SM代码	CPU_SM_1
说明	应用软件中的控制流监控
所有权	终端用户
具体实现	<p>对于永久故障，CPU核心的故障分布的主要部分和与程序计数器失控或终止直接相关的故障模式有关。它们的内在本质决定了不能通过标准软件测试方法（例如，SM_CPU_0）解决此类故障模式。因此，必须实现应用软件流的运行时间控制，以便监控和检测此类故障导致的与预期行为的偏差。将该机制链接到看门狗触发可确保能够检测到严重失控状态（或者在最坏情况下，程序计数器终止）。</p> <p>方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 充分记录并描述应用软件的不同内部状态（鼓励使用动态状态转换图）。 – 实现对应用软件每次不同状态间转换的正确性的监控。 – 检查异常应用软件程序循环期间所有预期状态之间的转换。 – 实现负责触发系统看门狗的功能，以便将触发（防止发生看门狗复位CPU的问题）限制在上述程序流监控方法的正确执行。 – 使用独立看门狗（IWDG）（或外部看门狗）的窗口功能有助于实现通过不同时钟源反馈的更稳健控制流机制。 <p>注： 安全指标均与使用的看门狗类型无关（采用独立或外部看门狗有助于缓解依赖性故障，参见第 4.2.2 节：时钟）</p>
错误报告	取决于实现
故障检测时间	取决于实现。看门狗超时时间间隔为上限。

表6. CPU_SM_1 (续)

已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	NA
多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	-

表7. CPU_SM_2

SM代码	CPU_SM_2
说明	应用软件中的双重计算
所有权	终端用户
具体实现	<p>考虑安全相关计算的时序冗余，以检测影响Arm® Cortex®-M3 CPU子部件（专用于数学计算和数据存取）的瞬时故障。</p> <p>方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 如果错误结果可能干扰系统安全功能，则需对此类安全相关的计算进行双重计算。因此，必须在原始应用软件源代码中仔细识别此类计算。 – 将数学运算和比较都当做计算。 – 使用原始数据的副本，并且如何可能的话使用等价公式进行第二次计算，来实现数学计算的冗余计算。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	瞬态
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	终端用户有责任小心地避免所用编译器优化功能的干预导致按照该使用条件引入的时序冗余被抵消。

表8. CPU_SM_3

SM代码	CPU_SM_3
说明	Arm® Cortex®-M3 HardFault异常
所有权	ST
具体实现	HardFault异常生成是一种在Arm® Cortex®-M3核心中实现的固有的安全机制，主要致力于拦截由于软件缺陷或软件设计中的错误导致的系统故障（例如，导致执行未定义的操作或未对齐的地址访问）。该安全机制还能检测CPU中导致上述异常操作的硬件随机故障。
错误报告	高优先级中断事件
故障检测时间	取决于具体实现，参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	无
周期性	连续
诊断测试	可以编写测试程序来验证HardFault异常的生成；总之，鉴于在硬件随机故障检测方面的预期作用不大，不建议进行此类实现。
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

表9. CPU_SM_4

SM代码	CPU_SM_4
说明	应用软件的堆栈加固
所有权	终端用户
具体实现	<p>为了解决影响CPU寄存器组的故障（主要是瞬时故障），需使用堆栈加固方法。该方法基于源代码修改，在寄存器传递至被调用函数的信息中引入信息冗余。</p> <p>方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 传递已传递参数值的冗余副本（可以对参数进行取反操作）并在函数中执行一致性检查。 – 传递已传递指针的冗余副本并在函数中执行一致性检查。 – 对于无冗余保护的参数，实施防御性编程技术（已传递值的真实性检查）。例如，将检查枚举字段的一致性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	该方法与适用于软件开发的IEC61508标准要求的防御性编程技术存在部分重叠。因此，如果应用软件的安全完整性等级高于或等于SC2，则可以优化。

表10. CPU_SM_5

SM代码	CPU_SM_5
说明	外部看门狗
所有权	终端用户
具体实现	使用链接到控制流监控方法（参见CPU_SM_1）的外部看门狗可解决CPU程序计数器或控制结构的故障模式。 可将外部看门狗设计为能够生成最终系统达到安全状态所需的信号组合。建议仔细检查第 3.3.1 节中列出的系统安全状态的相关假定要求。 它还有助于显著减少潜在的共因故障，因为外部看门狗有独立于STM32L0的时钟和供电。
错误报告	取决于实现
故障检测时间	取决于实现（看门狗超时间隔）
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	将从系统层面定义（合规项分析范围之外）
多重故障保护	CPU_SM_1：应用软件中的控制流监控
建议和已知限制	如果使用窗口看门狗，终端用户必须考虑应用软件执行中可能存在的容差，以避免伪错误报告（影响系统可用性）。

表11. CPU_SM_6

SM代码	CPU_SM_6
说明	独立看门狗
所有权	ST
具体实现	使用链接到控制流监控方法（参见CPU_SM_1）的IWDG看门狗可解决CPU程序计数器或控制结构的故障模式。
错误报告	复位信号生成
故障检测时间	取决于实现（看门狗超时间隔）
已解决故障模型	永久
取决于MCU配置	无
初始化	IWDG激活。建议在选项字节设置中使用“硬件看门狗”（在复位后自动使能IWDG）
周期性	连续
诊断测试	WDG_SM_1：启动时的看门狗软件测试
多重故障保护	CPU_SM_1：应用软件中的控制流监控 WDG_SM_0：配置寄存器的定期回读
建议和已知限制	IWDG干预能够达到潜在“不完全”的局部安全状态，因为它只能保证CPU复位，而不保证仍能继续执行应用软件以生成外部系统达到最终安全状态可能需要的输出信号组合。如果此限制不能被接受，则终端用户必须采用CPU_SM_5。

表12. CPU_SM_7

SM代码	CPU_SM_7
说明	MPU - 存储器保护单元
所有权	ST
具体实现	CPU存储器保护单元能够按照终端用户设定的标准检测对受保护存储区的非法访问。
错误报告	异常生成 (MemManage)
故障检测时间	参见功能文档
已解决故障模型	系统 (软件错误) 永久和瞬时 (仅程序计数器和存储器存取故障)
取决于MCU配置	无
初始化	应在启动时设定MPU寄存器
周期性	在线
诊断测试	不需要
多重故障保护	MPU_SM_0: 配置寄存器的定期回读
建议和已知限制	<p>当在应用软件中实现了多重安全功能时, 强烈建议使用存储分区并通过MPU功能提供保护。MPU具体可用于</p> <ul style="list-style-type: none"> – 执行权限规则 – 单独处理 – 执行访问规则 <p>MPU的硬件随机故障检测功能仅限于主要影响程序计数器和存储器访问CPU功能的少数几种故障模式。因此, 在STM32L0系列安全理念的框架内, 相关的诊断覆盖率预期将不那么重要。</p>

表13. MPU_SM_0

SM代码	MPU_SM_0
说明	MPU配置寄存器的定期回读
所有权	终端用户
具体实现	必须对MPU配置寄存器 (即使未被终端用户应用软件使用) 应用该方法。 关于实现该方法的详细信息, 可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

3.6.2 嵌入式FLASH

表14. FLASH_SM_0

SM代码	FLASH_SM_0
说明	闪存的定期软件测试
所有权	终端用户或ST
具体实现	影响系统闪存、存储单元和地址解码器的永久故障通过专用软件测试来解决，该测试使用基于签名的技术检查存储单元内容是否符合预期值。根据IEC 61508:2表A.5，此类技术的有效诊断覆盖率取决于签名相对于要保护的信息块长的宽度 - 因此，要仔细选择签名计算方法。简单签名方法（IEC 61508:7 - A.4.2改进校验和）是不够的，因为只能达到低覆盖率。 无需使用该测试处理信息块，因为在正常操作期间不使用信息块（无数据或程序提取）。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	闪存大小因产品编号而异
初始化	存储器签名也必须保存在闪存中
周期性	定期
诊断测试	可根据选择的测试实现设计策略，在软件中嵌入自诊断功能。
多重故障保护	CPU_SM_1：应用软件中的控制流监控 CPU_SM_0：核心定期自检软件
建议和已知限制	该测试预期会有较长的持续时间 - 因此必须考虑测试集成对应用软件执行的影响。 建议使用内部CRC模块。原则上，可以使用DMA功能进行数据传输。 <i>注： 可以从测试中排除未使用的闪存区。</i>

表15. FLASH_SM_1

SM代码	FLASH_SM_1
说明	应用软件中的控制流监控
所有权	终端用户
具体实现	影响系统闪存、存储单元和地址解码器的永久和瞬时故障可能干扰CPU的存取操作，导致错误的数据或指令提取。 通过在从闪存加载的应用软件中实现控制流监控技术，可以检测此类故障。 有关实现的详细信息，参见CPU_SM_1的说明。
错误报告	取决于实现
故障检测时间	取决于实现。看门狗超时时间间隔为上限。
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	NA

表15. FLASH_SM_1 (续)

多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	CPU_SM_1的正确实现取代了此要求。

表16. FLASH_SM_2

SM代码	FLASH_SM_2
说明	Arm® Cortex®-M3 HardFault异常
所有权	ST
具体实现	影响系统闪存（存储单元、地址解码器）的硬件随机故障（永久和瞬时）可导致错误的指令码提取，并最终产生Arm® Cortex®-M3 HardFault异常。参见CPU_SM_3获取详细说明。
错误报告	请参考CPU_SM_3
故障检测时间	请参考CPU_SM_3
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	请参考CPU_SM_3
周期性	连续
诊断测试	请参考CPU_SM_3
多重故障保护	请参考CPU_SM_3
建议和已知限制	无

表17. FLASH_SM_3

SM代码	FLASH_SM_3
说明	选项字写保护
所有权	ST
具体实现	该安全机制防止对选项字节的意外写入。使用该方法可以增强终端应用在发生系统故障时的稳健性。
错误报告	写保护异常
故障检测时间	不适用
已解决故障模型	无（仅系统故障）
取决于MCU配置	无
初始化	不需要（默认使能）
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	该方法解决软件应用中的系统故障，在解决运行时间内影响选项字节值的硬件随机故障方面功效为零。因此，没有相关的DC值。

表18. FLASH_SM_4

SM代码	FLASH_SM_4
说明	静态数据封装
所有权	终端用户
具体实现	如果静态数据保存在闪存中，必须实现使用编码功能（例如，CRC）通过校验和字段进行封装。 在使用静态数据之前，通过应用软件检查校验和有效性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

表19. FLASH_SM_5

SM代码	FLASH_SM_5
说明	具有负载验证的选项字节冗余
所有权	ST
具体实现	在每次上电复位后的选项字节加载期间，将进行选项字节的按位互补并验证其相应的互补选项字节。如不匹配，将报告错误。
错误报告	生成选项字节错误（OPTERR）
故障检测时间	不适用
已解决故障模型	永久
取决于MCU配置	无
初始化	无（始终使能）
周期性	启动
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

表20. FLASH_SM_6

SM代码	FLASH_SM_6
说明	未使用闪存区填充码
所有权	终端用户

表20. FLASH_SM_6 (续)

具体实现	未使用的闪存区必须填充确定数据。这种情况下，当程序计数器因影响CPU的瞬时故障跳出应用程序区时，系统将以确定的方式执行。
错误报告	NA
故障检测时间	NA
已解决故障模型	无（避错）
取决于MCU配置	无
初始化	NA
周期性	NA
诊断测试	NA
多重故障保护	NA
建议和已知限制	填充代码可以是NOP指令或会导致HardFault异常的非代码。

表21. FLASH_SM_8

SM代码	FLASH_SM_8
说明	读保护（RDP），写保护（WRP），专有代码读出保护（PCROP）
所有权	ST
具体实现	可使用这些保护功能保护闪存，防止非法读/写或擦除。通过这些技术与相关的不同保护级别的组合，终端用户能够构建有效的访问保护策略。
错误报告	参见功能文档 - 某些情况下会生成HardFault错误
故障检测时间	参见功能文档
已解决故障模型	系统
取决于MCU配置	无
初始化	不需要
周期性	连续
诊断测试	不需要
多重故障保护	不需要
建议和已知限制	闪存访问策略的硬件随机故障检测功能仅限于主要影响程序计数器和闪存接口功能的少数几种边缘故障模式。因此，在STM32L0系列安全理念的框架内，相关的诊断覆盖率预期将不那么重要。

3.6.3 内部 SRAM

表22. RAM_SM_0

SM代码	RAM_SM_0
说明	SRAM的定期软件测试
所有权	终端用户或ST
具体实现	为了提高SRAM数据单元的覆盖率并确保对影响地址解码器的永久故障的充分覆盖，需要对系统RAM存储器执行定期软件测试。算法的选择必须确保对RAM单元和地址解码器的目标SFF覆盖率。还必须收集所选方法覆盖有效性的证据。

表22. RAM_SM_0 (续)

错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	RAM大小因产品编号而异
初始化	取决于实现
周期性	定期
诊断测试	可根据选择的测试实现设计策略，在软件中嵌入自诊断功能。
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	<p>建议使用March测试C-。</p> <p>由于该测试属于破坏性测试，必须实现RAM内容恢复。还必须考虑测试执行期间可能发生的对中断服务例程的干扰（例如例程可能访问RAM中的无效内容）。</p> <p>注： 可以从测试中排除未使用的RAM区，由终端用户负责获取最终应用软件的实际情况。</p>

表23. RAM_SM_2

SM代码	RAM_SM_2
说明	应用软件的堆栈加固
所有权	终端用户
具体实现	<p>堆栈加固方法用于增强应用软件在发生影响地址解码器的SRAM故障时的稳健性。该方法基于源代码修改，在堆栈传递至被调用函数的信息中引入信息冗余。如果最终的应用软件结构与编译器设置之间的组合要求大量使用堆栈来传递函数参数，则方法的作用十分重要。</p> <p>实现过程与CPU_SM_4相同。</p>
错误报告	请参考CPU_SM_4
故障检测时间	请参考CPU_SM_4
已解决故障模型	请参考CPU_SM_4
取决于MCU配置	请参考CPU_SM_4
初始化	请参考CPU_SM_4
周期性	请参考CPU_SM_4
诊断测试	请参考CPU_SM_4
多重故障保护	请参考CPU_SM_4
建议和已知限制	请参考CPU_SM_4

表24. RAM_SM_3

SM代码	RAM_SM_3
说明	应用软件中安全相关变量的信息冗余
所有权	终端用户

表24. RAM_SM_3 (续)

具体实现	<p>为了解决影响SRAM控制器的瞬时故障，要求对保存在RAM中的安全相关系统变量实现信息冗余。</p> <p>该方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 充分识别并记录安全相关系统变量（就RAM读取故障导致的错误值会影响安全功能这一点而言）。 – 对于基于此类变量的算术计算或决策，执行两次并比较两个最终结果。 – 在两个冗余位置保存和更新安全相关变量，并在使用数据之前检查比较结果。 – 枚举字段必须使用非平凡值，每PST至少进行一次一致性检查 – 必须通过编码校验和（例如，CRC）保护SRAM中保存的数据向量
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	该安全方法的实现与Cortex®-M3的已预见方法（CPU_SM_1）存在部分重叠；因此，两种方法的实现都可以进行优化。

表25. RAM_SM_4

SM代码	RAM_SM_4
说明	应用软件中的控制流监控
所有权	终端用户
具体实现	<p>如果从SRAM执行终端用户应用软件，则影响存储器（存储单元和地址解码器）的永久和瞬时故障可能干扰程序执行。</p> <p>为了解决此类故障，需要实施该方法。</p> <p>有关实现的详细信息，参见CPU_SM_1的说明。</p>
错误报告	取决于实现
故障检测时间	取决于实现。看门狗超时间隔为上限。
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	NA
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	<p>仅在从SRAM执行应用软件时需要。</p> <p>CPU_SM_1的正确实现消除了此要求。</p>

表26. RAM_SM_5

SM代码	RAM_SM_5
说明	RAM中应用程序的定期完整性测试
所有权	终端用户
具体实现	如果在RAM中执行应用程序或诊断库，则需要保护代码本身的完整性，防止软错误损坏和相关代码突变。该方法必须定期通过校验和计算技术检查所保存代码的完整性（每PST至少一次）。关于实现详情，参见类似方法FLASH_SM_0。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	可根据选择的测试实现设计策略，在软件中嵌入自诊断功能。
多重故障保护	CPU_SM_0：核心定期自检软件 CPU_SM_1：应用程序中的控制流监控
建议和已知限制	只需要在从RAM执行应用程序或诊断库时实现该方法。

3.6.4 系统总线架构

表27. BUS_SM_0

SM代码	BUS_SM_0
说明	互连的定期软件测试
所有权	终端用户
具体实现	需定期测试片内连接资源（总线矩阵、AHB或APB桥）进行永久故障检测。请注意，STM32L0系列MCU没有用于保护这些结构的硬件安全机制。测试执行这些共享资源的连接测试，包括外设之间仲裁机制的测试。 根据IEC 61508:2表A.8提到的A.7.4，该方法被认为能够达到高覆盖率。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	可将实现视为与一些外设要求的广泛使用的“配置寄存器的定期回读”存在大部分重叠。

表28. BUS_SM_1

SM代码	BUS_SM_1
说明	片内数据交换中的信息冗余
所有权	终端用户
具体实现	该方法要求为MCU内部交换的每条数据消息添加某种冗余（例如，数据包层面的CRC校验和）。 在使用数据之前，通过应用软件使用校验和验证消息完整性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	实现可能与要求为通信外设提供数据消息信息冗余的其他安全机制存在大部分重叠。因此，可以进行优化。

表29. LOCK_SM_0

SM代码	LOCK_SM_0
说明	配置选项的锁定机制
所有权	ST
具体实现	STM32L0系列MCU提供扩展保护，防止某些外设和系统寄存器发生意外的配置更改（例如，PVD_LOCK、定时器）；扩展保护检测软件应用中的系统故障。使用该方法可增强终端应用在发生系统故障时的稳健性。
错误报告	未生成（被锁定时，将忽略寄存器重写）
故障检测时间	NA
已解决故障模型	无（仅系统故障）
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	不需要
建议和已知限制	无相关DC，因为该测试处理系统故障。

3.6.5 NVIC和EXTI控制器

表30. NVIC_SM_0

SM代码	NVIC_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	<p>通过定期检查配置寄存器以确定是否有系统外设不符合其预期值来实现该测试。预期值之前被保存在RAM中，并在每次配置更改后及时更新。该方法主要通过检测寄存器内容中的位翻转来解决影响配置寄存器的瞬时故障。它还解决寄存器上的永久故障，因为会在外设更新后在PST中至少执行一次该方法。</p> <p>对于在设置错误时其内容可能干扰NVIC或EXTI行为的任何配置寄存器，都必须实现该方法。检查对象包括NVIC向量表。</p> <p>根据先进的汽车安全标准ISO26262，该方法可获得高诊断覆盖率（参见ISO26262:5，表D.4）</p> <p>在签名理念的基础上，可以实现一种对SRAM空间要求更低的替代性有效实现：</p> <ul style="list-style-type: none"> – 连续读取要检查的外设寄存器，计算CRC校验和（鼓励使用硬件CRC） – 将获得的签名与标准值进行比较（在每次寄存器更新后以相同方式计算，并保存在SRAM中） – 通过应用软件检查签名的一致性 - 将签名不匹配视为检测失败。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	必须在启动后执行首次检查之前读取配置寄存器的值。
周期性	定期
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	<p>该方法只解决影响配置寄存器的故障，而不是外设核心逻辑或外部接口。</p> <p>必须注意包含配置与状态位的混合组合的寄存器。在保存影响签名的寄存器内容之前，必须使用掩码，并执行相关检查以避免错误的检测。</p>

表31. NVIC_SM_1

SM代码	NVIC_SM_1
说明	预期和意外中断检查
所有权	终端用户

表31. NVIC_SM_1 (续)

具体实现	<p>根据IEC 61508:2表A.1的建议，必须实现针对中断连续、缺失或交叉的诊断措施。在应用软件层面实现预期和意外中断检查方法。</p> <p>方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 充分记录MCU的已实现中断列表，并尽可能报告每个请求的预期频率（例如与ADC转换完成相关的中断，因此是一种确定的方式）。 – 为服务的每种中断请求提供单独的计数器，以便检测给定时间帧内：a)没有中断请求产生；b)过多的中断请求（“babbling idiot”中断源）。必须按照具体的中断预期频率调整对时间帧持续时间的控制。 – 与未使用的中断源相关的中断向量指向默认处理程序，它会在触发时报告故障情况（意外中断）。 – 如果不同的源之间共享中断服务例程，则实现对调用者身份的真实性检查。 – 使用此处描述的相同方法处理与非安全相关外设相关的中断请求，不考虑它们的发起者安全分类。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	为了降低方法实现的复杂性，建议使用轮询技术（如可能）而不是中断来实现终端系统。

3.6.6 直接存储器访问控制器 (DMA)

表32. DMA_SM_0

SM代码	DMA_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	<p>必须对DMA配置寄存器和通道地址寄存器应用该方法。</p> <p>关于实现该方法的详细信息，可在第 3.6.5 节中找到。</p>
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表33. DMA_SM_1

SM代码	DMA_SM_1
说明	通过DMA传输的数据包的信息冗余
所有权	终端用户
具体实现	通过为DMA传输的数据包添加冗余检查（例如CRC或类似检查）来实现该方法。完整数据包冗余是一种过度行为。 校验和编码功能必须足够稳健，以保证数据包中单个位反转的发现概率至少为90% 在使用数据之前，必须通过应用软件检查数据包的一致性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	举例来说，对于校验和的编码功能，只使用逐位相加是不合适的。

表34. DMA_SM_2

SM代码	DMA_SM_2
说明	通过DMA传输的数据包的信息冗余，包括发送者和接收者标识符
所有权	终端用户
具体实现	该方法有助于识别MCU内部通过DMA交换的消息的来源和发起者。 通过为受保护消息添加额外字段来实现，MCU层面有固定的编码约定用于识别消息类型。 识别字段的指导原则如下： – DMA事务的每一对可能的发送者或接收者必须具有不同的识别字段值 – 选择的值必须是枚举型非平凡值 – 在使用数据之前，通过应用软件检查识别字段值与消息类型是否相符。 当与DMA_SM_4一起实现时，该方法在源和目标实体之间提供一种“虚拟通道”。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

表35. DMA_SM_3

SM代码	DMA_SM_3
说明	DMA的定期软件测试
所有权	终端用户
具体实现	<p>该方法要求定期测试DMA基础功能，通过从一个源到另一个源（例如，从存储器到存储器）的数据包确定传输和在目标上检查消息传输的正确性来实现。数据包由非平凡模式构成（避免使用值0x0000、0xFFFF），并且其组织方式允许在检查以下故障期间进行检测：</p> <ul style="list-style-type: none"> – 不完全打包传输 – 单个传输字中发生错误 – 打包传输数据中的顺序错误
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

表36. DMA_SM_4

SM代码	DMA_SM_4
说明	DMA事务感知
所有权	终端用户
具体实现	<p>DMA事务是非确定性事务，因为它通常由外部事件驱动，例如通信消息接收。总之，设计周密的安全系统应尽可能全面地控制事件 - 例如可参考IEC61508:3表格2第13项要求了解软件架构。</p> <p>该方法建立在系统已知预期DMA事务的频率和类型的基础上（例如，外接传感器应定期向STM32外设发送某些消息）。通过专用状态机监控DMA事务，将能够检测缺失或意外的DMA活动。</p>
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	由于DMA事务终止通常与中断生成有关，可将该方法的实现与安全机制NVIC_SM_1合并。

3.6.7 通用同步收发器（USART1/2/4/5）、低功耗通用异步收发器（LPUART1）

表37. UART_SM_0

SM代码	UART_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对UART配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表38. UART_SM_1

SM代码	UART_SM_1
说明	协议错误信号
所有权	ST
具体实现	USART通信模块内置协议错误检查（例如，额外校验位检查、上溢和帧错误），用于检测网络相关异常情况。总之，这些机制能够检测影响模块本身的硬件随机故障的边际百分比。 通常在标准通信软件中处理连接到这些检查器的错误信号，从而减少开销。
错误报告	生成错误标志和可选中断事件
故障检测时间	取决于外设配置（例如，波特率），参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	UART_SM_2：消息的信息冗余技术
建议和已知限制	USART通信模块具备多种不同配置 - 通信错误检查的实际构成取决于选择的配置。

表39. UART_SM_2

SM代码	UART_SM_2
说明	消息的信息冗余技术
所有权	终端用户
具体实现	通过使用编码功能为UART传输的数据包添加冗余检查（例如CRC或类似检查）来实现该方法。校验和编码功能必须足够稳健，以保证数据包中单个位反转的发现概率至少为90%。 在使用数据之前，必须通过应用软件检查数据包的一致性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	假定UART通信的另一方具有执行所述检查的同等能力。 不应使用传输完全冗余（消息重复），因为其检测能力仅限于通信单元故障模式的一个子集。 举例来说，对于校验和的编码功能，只使用逐位相加是不合适的。

表40. UART_SM_3

SM代码	UART_SM_3
说明	消息的信息冗余技术，包括端到端安全
所有权	终端用户
具体实现	该方法旨在保护外设与其外部通信对象之间的通信。 参见CAN_SM_2说明获取详细信息。
错误报告	请参考CAN_SM_2
故障检测时间	请参考CAN_SM_2
已解决故障模型	请参考CAN_SM_2
取决于MCU配置	请参考CAN_SM_2
初始化	请参考CAN_SM_2
周期性	请参考CAN_SM_2
诊断测试	请参考CAN_SM_2
多重故障保护	请参考CAN_SM_2
建议和已知限制	<i>注：</i> 假定UART通信的另一方具有执行所述检查的同等能力。参见CAN_SM_2获取更多说明。

3.6.8 内部集成电路（I2C1/2）

表41. IIC_SM_0

SM代码	IIC_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对I2C配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表42. IIC_SM_1

SM代码	IIC_SM_1
说明	协议错误信号
所有权	ST
具体实现	I2C通信模块内置协议错误检查（例如，上溢、下溢、数据包等），用于检测网络相关异常情况。总之，这些机制能够检测影响模块本身的硬件随机故障的边际百分比。
错误报告	生成错误标志和可选中断事件
故障检测时间	取决于外设配置（例如，波特率），参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	IIC_SM_2：消息的信息冗余技术
建议和已知限制	采用SMBus选项将授权激活更高效的协议层面硬件检查，例如CRC-8数据包保护。

表43. IIC_SM_2

SM代码	IIC_SM_2
说明	消息的信息冗余技术
所有权	终端用户
具体实现	通过使用编码功能为I2C传输的数据包添加冗余检查（例如CRC或类似检查）来实现该方法。校验和编码功能必须足够稳健，以保证数据包中单个位反转的发现概率至少为90%。 在使用数据之前，必须通过应用软件检查数据包的一致性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	假定I2C通信的另一方具有执行所述检查的同等能力。 不应使用传输完全冗余（消息重复），因为其检测能力仅限于通信单元故障模式的一个子集。 举例来说，对于校验和的编码功能，只使用逐位相加是不合适的。 如果能够通过硬件插入CRC，则以IIC_SM_3取代该方法。

表44. IIC_SM_3

SM代码	IIC_SM_3
说明	CRC数据包层面
所有权	ST
具体实现	I2C通信模块能够为特定操作模式（SMBus）激活向包数据自动插入（和检查）CRC校验和。
错误报告	生成错误标志和可选中断事件
故障检测时间	取决于外设置（例如，波特率），参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	IIC_SM_2：消息的信息冗余技术
建议和已知限制	无

表45. IIC_SM_4

SM代码	IIC_SM_4
说明	消息的信息冗余技术，包括端到端安全
所有权	终端用户
具体实现	该方法旨在保护I2C外设与其外部通信对象之间的通信。 参见CAN_SM_2说明获取详细信息。
错误报告	请参考CAN_SM_2
故障检测时间	请参考CAN_SM_2
已解决故障模型	请参考CAN_SM_2
取决于MCU配置	请参考CAN_SM_2
初始化	请参考CAN_SM_2
周期性	请参考CAN_SM_2
诊断测试	请参考CAN_SM_2
多重故障保护	请参考CAN_SM_2
建议和已知限制	重要说明：假定I2C通信的另一方具有执行所述检查的同等能力。 参见CAN_SM_2获取更多说明。

3.6.9 串行外设接口（SPI1/2）

表46. SPI_SM_0

SM代码	SPI_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对SPI配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表47. SPI_SM_1

SM代码	SPI_SM_1
说明	协议错误信号
所有权	ST
具体实现	SPI通信模块内置协议错误检查（例如，上溢、下溢、超时等），用于检测网络相关异常情况。总之，这些机制能够检测影响模块本身的硬件随机故障的边际百分比。
错误报告	生成错误标志和可选中断事件
故障检测时间	取决于外设配置（例如，波特率），参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	NA
多重故障保护	SPI_SM_2：消息的信息冗余技术
建议和已知限制	无

表48. SPI_SM_2

SM代码	SPI_SM_2
说明	消息的信息冗余技术
所有权	终端用户
具体实现	通过使用编码功能为SPI传输的数据包添加冗余检查（例如CRC或类似检查）来实现该方法。校验和编码功能必须足够稳健，以保证数据包中单个位反转的发现概率至少为90%。 在使用数据之前，必须通过应用软件检查数据包的一致性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	假定SPI通信的另一方具有执行所述检查的同等能力。 不应使用传输完全冗余（消息重复），因为其检测能力仅限于通信单元故障模式的一个子集。 举例来说，对于校验和的编码功能，只使用逐位相加是不合适的。 如果能够通过硬件插入CRC，则以SPI_SM_3取代该方法。

表49. SPI_SM_3

SM代码	SPI_SM_3
说明	CRC数据包层面
所有权	ST
具体实现	SPI通信模块能够激活向包数据自动插入（和检查）CRC-8或CRC-18校验和。
错误报告	生成错误标志和可选中断事件
故障检测时间	取决于外设配置（例如，波特率），参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	SPI_SM_2：消息的信息冗余技术
建议和已知限制	无

表50. SPI_SM_4

SM代码	SPI_SM_4
说明	消息的信息冗余技术，包括端到端安全
所有权	终端用户
具体实现	该方法旨在保护SPI外设与其外部通信对象之间的通信。 参见CAN_SM_2说明获取详细信息。
错误报告	请参考CAN_SM_2
故障检测时间	请参考CAN_SM_2
已解决故障模型	请参考CAN_SM_2
取决于MCU配置	请参考CAN_SM_2
初始化	请参考CAN_SM_2
周期性	请参考CAN_SM_2
诊断测试	请参考CAN_SM_2
多重故障保护	请参考CAN_SM_2
建议和已知限制	重要说明：假定SPI通信的另一方具有执行所述检查的同等能力。 参见CAN_SM_2获取更多说明。

3.6.10 USB - 2.0通用串行总线接口FS模块

表51. USB_SM_0

SM代码	USB_SM_0
说明	配置寄存器的定期回读
所有权	终端用户

表51. USB_SM_0 (续)

具体实现	必须对USB配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表52. USB_SM_1

SM代码	USB_SM_1
说明	协议错误信号
所有权	ST
具体实现	USB通信模块内置协议错误检查（例如，上溢、下溢、NRZI、比特填充等），用于检测网络相关异常情况。总之，这些机制能够检测影响模块本身的硬件随机故障的边际百分比。
错误报告	生成错误标志和可选中断事件
故障检测时间	取决于外设配置（例如，波特率），参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	USB_SM_2：消息的信息冗余技术
建议和已知限制	无

表53. USB_SM_2

SM代码	USB_SM_2
说明	消息的信息冗余技术
所有权	ST或终端用户
具体实现	实现所需消息信息冗余，USB通信模块具备硬件功能。它本质上能够激活向包数据自动插入（和检查）CRC校验和。
错误报告	生成错误标志和可选中断事件
故障检测时间	取决于外设配置（例如，波特率），参见功能文档

表53. USB_SM_2 (续)

已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	错误报告配置（如果计划了中断事件）
周期性	连续
诊断测试	不需要
多重故障保护	USB_SM_2: 消息的信息冗余技术
建议和已知限制	无

表54. USB_SM_3

SM代码	USB_SM_3
说明	消息的信息冗余技术，包括端到端安全
所有权	终端用户
具体实现	该方法旨在保护USB外设与其外部通信对象之间的通信。 参见CAN_SM_2说明获取详细信息。
错误报告	请参考CAN_SM_2
故障检测时间	请参考CAN_SM_2
已解决故障模型	请参考CAN_SM_2
取决于MCU配置	请参考CAN_SM_2
初始化	请参考CAN_SM_2
周期性	请参考CAN_SM_2
诊断测试	请参考CAN_SM_2
多重故障保护	请参考CAN_SM_2
建议和已知限制	当使用了USB批量或同步传输时，该方法适用。对于其他传输模式，USB硬件协议已实现该要求的多个特性。 参见CAN_SM_2获取更多说明。

3.6.11 触摸感应控制器（TSC）

表55. TSC_SM_0

SM代码	TSC_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对TSC配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0

表55. TSC_SM_0 (续)

初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表56. TSC_SM_1

SM代码	TSC_SM_1
说明	通过应用软件进行多重采集
所有权	终端用户
具体实现	该方法通过对TSC输入数据执行多重采集来实现时序信息冗余。然后，使用多重采集数据确定采集的正确状态。 该方法与事件计数TSC模块的原生特性相重叠，目的是确保稳定采集，不受外部噪声干扰。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	瞬态
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

表57. TSC_SM_2

SM代码	TSC_SM_2
说明	TSC采集永久故障的应用层面检测
所有权	终端用户
具体实现	该方法必须检测导致触摸感应事件采集错误或缺失的TSC模块永久故障。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	不需要

表57. TSC_SM_2 (续)

多重故障保护	CPU_SM_0: 核心定期自测软件
建议和已知限制	由于该解决方案严格取决于应用, 此处未给出其实现方式的详细指导原则。由于解决方案不能完全基于微控制器资源, 必须利用最终系统的其他元件来实现。

3.6.12 模数转换器 (ADC)

表58. ADC_SM_0

SM代码	ADC_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对ADC配置寄存器应用该方法。 关于实现该方法的详细信息, 可在 第 3.6.5 节 中找到
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表59. ADC_SM_1

SM代码	ADC_SM_1
说明	通过应用软件进行多重采集
所有权	终端用户
具体实现	该方法通过对同一输入信号执行多重采集来实现时序信息冗余。然后, 通过滤波器算法组合多重采集数据以确定信号的正确值。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	通常通过终端用户应用软件的设计来满足该建议。在工业应用中, 在使用多重采集后进行平均值操作是一种常用技术, 用于克服传感器线上的杂散EMI干扰。

表60. ADC_SM_2

SM代码	ADC_SM_2
说明	通过应用软件进行范围检查
所有权	终端用户
具体实现	<p>方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 调查并充分记录要获取的数据的预期范围。请注意，在设计良好的应用中，正常操作期间的输入信号不大可能十分接近或超过上轨和下轨限值（信号采集饱和）。 – 如果应用软件识别出系统状态，将在范围检查实现中使用该信息。例如，如果ADC值是通过功率负载的电流测量值，读取到异常值（例如电流与负载供应反向）可能表示采集模块中存在故障。 – 由于ADC模块是可能有的不同外部源之间共享的，组合采集的不同信号的真实性检查可能有助于以十分高效的方式覆盖整个输入范围。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	该安全机制的实现（和相关诊断效率）严重依赖于应用。

表61. ADC_SM_3

SM代码	ADC_SM_3
说明	ADC的定期软件测试
所有权	终端用户
具体实现	<p>该方法的实现方式是采集多个信号并将读出值与已知的预期值进行比较。方法实现可能有不同复杂度级别：</p> <ul style="list-style-type: none"> – 基础复杂度：高电平或低电平（V_{DD}或V_{SS}）以及内部参考电压的采集与检查 – 高复杂度：除了基础复杂度测试，还采集连接到ADC输入的DAC输出并检查所有电压偏移和线性度
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	可使用两种不同复杂度方法的组合来更好地优化高需求安全功能的测试频率。

表62. ADC_SM_4

SM代码	ADC_SM_4
说明	ADC输入的1oo2方案
所有权	终端用户
具体实现	该安全机制的实现方法是，使用两个不同SAR ADC通道采集相同输入信号。应用软件检查两个读数是否一致。 建议使用属于不同ADC模块的两个不同ADC通道。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	ADC_SM_0：ADC配置寄存器的定期回读
建议和已知限制	该方法可与ADC_SM_0ADC_SM_2ADC_SM_3结合使用，以达到最高ADC模块诊断覆盖率。

3.6.13 数模转换器（DAC）

表63. DAC_SM_0

SM代码	DAC_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对DAC配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表64. DAC_SM_1

SM代码	DAC_SM_1
说明	ADC通道上的DAC输出环回
所有权	终端用户
具体实现	通过将激活的DAC输出路由到一个ADC通道并检查输出电流值是否符合其预期值来实现。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续或按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	处理瞬时故障的效率与最终应用的特性有关。我们将Tm定义为触发相关安全功能所需的DAC错误信号的最短持续时间。执行测试频率高于1/Tm时效率最高。

3.6.14 比较器（COMP）

表65. COMP_SM_0

SM代码	COMP_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对COMP配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表66. COMP_SM_1

SM代码	COMP_SM_1
说明	比较器的1oo2方案
所有权	终端用户
具体实现	该安全机制的实现方法是，使用两个内部比较器做出相同决策。这要求对比较器投票进行相应处理。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	该方法与“窗口”比较器功能不兼容。

表67. COMP_SM_2

SM代码	COMP_SM_2
说明	输入的真实性检查
所有权	终端用户
具体实现	该方法用于在专用ADC通道上冗余采集比较器功能的模拟输入，并通过测量值定期检查比较器输出的一致性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

表68. COMP_SM_3

SM代码	COMP_SM_3
说明	通过应用软件进行多重采集
所有权	终端用户
具体实现	该方法要求应用软件不按照比较器单次转换进行决策，而是在多个事件后或比较器触发条件存在特定的时间段后进行决策。

表68. COMP_SM_3 (续)

错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	瞬态
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	通常在终端用户应用上通过设计满足该建议 - 在工业应用中, 多重采集并取平均值是一种常用技术, 用于克服传感器线上的杂散EMI干扰。

表69. COMP_SM_4

SM代码	COMP_SM_4
说明	比较器锁定机制
所有权	ST
具体实现	该安全机制防止比较器控制和状态寄存器发生配置变化; 因此, 它解决软件应用中的系统故障。
错误报告	NA
故障检测时间	NA
已解决故障模型	无 (避错)
取决于MCU配置	无
初始化	必须使用COMP_CSR寄存器中的COMPxLOCK位使能锁定保护。
周期性	连续
诊断测试	NA
多重故障保护	NA
建议和已知限制	该方法不解决软错误导致的比较器配置变化。

3.6.15 基本定时器 (TIM 6/7)

表70. GTIM_SM_0

SM代码	GTIM_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对通用计数器定时器TIM6或TIM7配置寄存器应用该方法。 关于实现该方法的详细信息, 可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0

表70. GTIM_SM_0 (续)

已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表71. GTIM_SM_1

SM代码	GTIM_SM_1
说明	计数定时器的1oo2
所有权	终端用户
具体实现	<p>该方法通过软件在两个计数资源之间实现1oo2方案。</p> <p>方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 使用相同时基或频率设定两个定时器。 – 对于用作时基的定时器：在应用软件中使用一个定时器作为时基源，另一个仅用于检查。在应用层面执行1oo2的一致性检查，在每次使用定时器值时比较两个计数器值以影响安全功能。 – 在中断生成时使用：使用第一个定时器作为服务例程的主要中断源，并使用第二个定时器作为中断例程开始时要检查的“参考”。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	为避免得到错误的诊断结果，建议在定时器检查中实现容差。

3.6.16 高级、通用和低功耗定时器（TIM/2/3/21/22，LPTIM1/2）

注：由于定时器配备许多不同通道，这些通道彼此独立，并能设定用于实现不同功能，因此要为每个通道单独选择安全机制。

表72. ATIM_SM_0

SM代码	ATIM_SM_0
说明	配置寄存器的定期回读
所有权	终端用户

表72. ATIM_SM_0 (续)

具体实现	必须对高级、通用和低功耗定时器TIM1/2/3/4/5/8/15/16/17 LPTIM1/2配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表73. ATIM_SM_1

SM代码	ATIM_SM_1
说明	计数定时器的1oo2
所有权	终端用户
具体实现	该方法通过软件在两个计数资源之间实现1oo2方案。 方法实现的指导原则如下： <ul style="list-style-type: none"> – 使用相同时基或频率设定两个定时器。 – 对于用作时基的定时器：在应用软件中使用一个定时器作为时基源，另一个仅用于检查。在应用层面执行1oo2的一致性检查，在每次使用定时器值时比较两个计数器值以影响安全功能。 – 在中断生成时使用：使用第一个定时器作为服务例程的主要中断源，并使用第二个定时器作为中断例程开始时要检查的“参考”。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	为避免得到错误的诊断结果，建议在定时器检查中实现容差。 该方法适用于只用作越时计数器的定时器通道。

表74. ATIM_SM_2

SM代码	ATIM_SM_2
说明	输入捕获定时器的1002
所有权	终端用户
具体实现	该方法设计用于保护用于外部信号捕获和测量（例如“输入捕获”和“编码器读取”）的定时器。方法实现要求将外部信号也连接到冗余定时器，并在应用层面对测得的数据执行一致性检查。 定时器之间的一致性检查将在每次应用软件使用读数时执行。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	为减少共因故障的潜在影响，建议对属于不同定时器模块并映射到器件封装上非相邻引脚的通道使用冗余检查。

表75. ATIM_SM_3

SM代码	ATIM_SM_3
说明	PWM输出的环回方案
所有权	终端用户
具体实现	通过将PWM连接到单独的定时器通道以获取生成的波形特性来实现该方法。 指导原则如下： <ul style="list-style-type: none"> – 测量PWM频率和占空比并检查是否符合预期值。 – 为减少共因故障的潜在影响，建议对属于不同定时器模块并映射到器件封装上非相邻引脚的通道使用环回检查。 该措施可以由终端用户负责用最终应用中已有的评级相同的其他环回方案来取代。例如，如果使用PWM驱动外部功率负载，可以使用在线电流值的读数而不是PWM占空比测量值。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要

表75. ATIM_SM_3 (续)

多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	处理瞬时故障的效率与最终应用的特性有关。我们将Tm定义为触发相关安全功能所需的PWM错误信号（错误频率、错误占空比或二者兼有）的最短持续时间。执行测试频率高于1/Tm时效率最高。

表76. ATIM_SM_4

SM代码	ATIM_SM_4
说明	定时器的锁定保护
所有权	ST
具体实现	该安全机制使终端用户能够锁定指定的配置选项，避免应用软件进行意外修改。因此，它解决软件开发的系统故障。
错误报告	NA
故障检测时间	NA
已解决故障模型	无（避错）
取决于MCU配置	无
初始化	必须使用TIMx_BDTR寄存器中的LOCK位使能锁定保护。
周期性	连续
诊断测试	NA
多重故障保护	NA
建议和已知限制	该方法不解决软错误导致的定时器配置变化。

注： 这里没有单独提及IRTIM，它主要通过TIM16和TIM17功能实现。请参见相关指示。

3.6.17 通用输入/输出（GPIO）- 端口A/B/C/D/E/F/G/H

表77. GPIO_SM_0

SM代码	GPIO_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对GPIO配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	GPIO可用性因产品编号而异
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0

表77. GPIO_SM_0 (续)

多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表78. GPIO_SM_1

SM代码	GPIO_SM_1
说明	输入GPIO线的1oo2
所有权	终端用户
具体实现	该方法适用于将GPIO线用作输入的情况。通过将外部的安全相关信号连接到两个独立的GPIO线来实现。在每次使用该信号的输入时，应用软件比较两个GPIO值。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	为了减少共因故障的潜在影响，建议使用GPIO线： <ul style="list-style-type: none"> – 属于不同I/O端口（例如，端口A和B） – 具有不同位号（例如，PORTA.1和PORTB.5） – 映射到器件封装上的不相邻引脚

表79. GPIO_SM_2

SM代码	GPIO_SM_2
说明	输出GPIO线的环回模式
所有权	终端用户
具体实现	该方法适用于将GPIO线用作输出的情况。通过环回方案（将输出连接到设定为输入的不同GPIO线）和使用输入线检查输出端口的预期值来实现。应用软件定期以及每次更新输出时执行比较。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要

表79. GPIO_SM_2 (续)

多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	<p>为了减少共因故障的潜在影响, 建议使用GPIO线:</p> <ul style="list-style-type: none"> – 属于不同I/O端口 (例如, 端口A和B) – 具有不同位号 (例如, PORTA.1和PORTB.5) – 映射到器件封装上的不相邻引脚 <p>处理瞬时故障的效率与最终应用的特性有关。我们将T_m定义为触发相关安全功能所需的GPIO输出错误信号的最短持续时间。执行测试频率高于1/T_m时效率最高。</p>

表80. GPIO_SM_3

SM代码	GPIO_SM_3
说明	GPIO 端口配置锁定寄存器
所有权	ST
具体实现	该安全机制防止GPIO寄存器发生配置变化; 因此, 它解决软件应用中的系统故障。鼓励使用该方法增强终端应用在发生系统故障时的稳健性。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	无 (仅系统故障)
取决于MCU配置	无
初始化	在应用软件写入最终的GPIO配置后, 必须对GPIOx_LCKR的位16 (LCKK) 应用正确的写序列。
周期性	连续
诊断测试	不需要
多重故障保护	不需要
建议和已知限制	该方法不解决可能在运行时间导致GPIO寄存器上位翻转的瞬时故障 (软错误)。

3.6.18 实时时钟模块 (RTC)

表81. RTC_SM_0

SM代码	RTC_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	<p>必须对RTC配置寄存器应用该方法。</p> <p>关于实现该方法的详细信息, 可在第 3.6.5 节中找到。</p>
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0

表81. RTC_SM_0 (续)

诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表82. RTC_SM_1

SM代码	RTC_SM_1
说明	运行RTC的应用检查
所有权	终端用户
具体实现	<p>应用软件对RTC日历或时间数据实施某种真实性检查，主要在上电后和RTC执行后续日期读取后。</p> <p>方法实现的指导原则如下：</p> <ul style="list-style-type: none"> – 使用RTC备份寄存器保存编码信息，以便检测掉电期间是否缺失VBAT。 – 使用RTC备份寄存器按当前日期或时间定期保存压缩信息 – 应用软件在上电后执行最低限度的日期读数一致性检查（检测“过去”日期或时间检索）。 – 应用软件定期检查RTC是否真正在运行，方法是读取RTC时间戳进度并与基于STM32内部时钟或定时器的时间测量值进行比较。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	该方法为RTC故障模式提供有限的诊断覆盖率。对于RTC时间戳精度可能严重影响安全功能（例如，医疗数据存储设备）的终端用户应用，强烈建议采用更有效的系统层面措施。

表83. RTC_SM_2

SM代码	RTC_SM_2
说明	备份寄存器的信息冗余
所有权	终端用户
具体实现	<p>必须通过具有编码功能的校验和（例如，CRC）保护RTC备份寄存器中保存的数据。在使用保存的数据之前，应用软件必须检查校验和。</p> <p>该方法将保证数据不会因备用电池故障而被擦除。</p>
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无

表83. RTC_SM_2 (续)

初始化	取决于实现
周期性	按需
诊断测试	不需要
多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	该安全方法的实现与Cortex®-M3的已预见方法 (CPU_SM_1) 存在部分重叠; 因此, 两种方法的实现都可以进行优化。

电源控制

表84. RTC_SM_3

SM代码	RTC_SM_3
说明	用于检测时间戳/事件捕获中的故障的应用层面措施
所有权	终端用户
具体实现	该方法必须能够检测影响RTC功能的故障, 以便正确执行时间戳/事件捕获功能。由于该解决方案严格取决于应用, 此处未给出其实现方式的详细指导原则。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	定期/按需
诊断测试	不需要
多重故障保护	CPU_SM_0: 核心定期自测软件
建议和已知限制	仅当在安全功能实现中使用了时间戳/事件捕获功能时, 才必须使用该方法。值得注意的是, 假定要求ASR7 (参考 第 3.3.1 节) 禁止在MCU处于睡眠或停止模式的安全相关应用中使用时间戳/事件捕获。

表85. VSUP_SM_0

SM代码	VSUP_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对配置寄存器应用该方法。 关于实现该方法的详细信息, 可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0

表85. VSUP_SM_0 (续)

多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表86. VSUP_SM_1

SM代码	VSUP_SM_1
说明	电源电压内部监控 (PVD)
所有权	ST
具体实现	该器件还有一个嵌入式可编程电压检测器 (PVD)，用于监视 V_{DD} 电源并将其与 V_{PVD} 阈值进行比较。当 V_{DD} 低于 V_{PVD} 阈值或 V_{DD} 高于 V_{PVD} 阈值时，将产生中断。
错误报告	中断事件生成
故障检测时间	取决于阈值设定，参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	通过功率控制寄存器 (PWR_CR) 中的PVDE位和阈值设定使能保护
周期性	连续
诊断测试	VSUP_SM_0: 配置寄存器的定期回读
多重故障保护	CPU_SM_5: 外部看门狗
建议和已知限制	内部监控PVD处理与错误 V_{DD} 值相关的故障的能力有限。如果实现特定的功耗模式，可以通过VSUP_SM_4 (PVM) 集成 (即，将某些独立电源直接连接到 V_{DD} ，以便使用PVD和PVM控制 V_{DD} 的高和低阈值)。 内部监控PVD解决影响STM32L0系列内部调压器的故障的能力有限。参见设备FMEA了解详情。

表87. VSUP_SM_2

SM代码	VSUP_SM_2
说明	独立看门狗
所有权	ST
具体实现	独立看门狗通过 V_{DD} 直接供电；因此，数字逻辑 (核心或外设) 电源的重大故障不会影响其行为，但可能导致IWDG超时违规。
错误报告	复位信号生成
故障检测时间	取决于实现 (看门狗超时时间)
已解决故障模型	永久
取决于MCU配置	无
初始化	IWDG激活。建议在选项字节设置中使用“硬件看门狗” (在复位后自动使能IWDG)
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_1: 应用软件中的控制流监控
建议和已知限制	外部看门狗 (参见CPU_SM_5) 可保证相同保护级别。

表88. VSUP_SM_3

SM代码	VSUP_SM_3
说明	内部温度传感器检查
所有权	终端用户
具体实现	必须定期测试内部温度传感器，以便检测模温的异常上升 - 电源电压系统中的硬件故障可能导致功耗过高，进而引起温度上升。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	无
周期性	定期
诊断测试	不需要
多重故障保护	VSUP_SM_3: 电源电压内部监控 (PVD)
建议和已知限制	该方法还降低了影响MCU和导致温度过高的共因发生的可能性。 参见数据手册以设置温度阈值。

表89. VSUP_SM_4

SM代码	VSUP_SM_4
说明	外设电压监测 (PVM)
所有权	ST
具体实现	该器件还有一个嵌入式可编程电压检测器 (PVM)，用于监控3个独立电源并将其与阈值进行比较。当独立电源电压低于低阈值或高于高阈值时，会生成中断。
错误报告	特定EXTI线上生成中断
故障检测时间	取决于阈值设定，参见功能文档
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	在电源控制寄存器中对所选电源轨执行保护使能和阈值设定。
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_5: 外部看门狗
建议和已知限制	该方法可与VSUP_SM_0结合使用，以实现V _{DD} 值的全面监控。

3.6.19 复位和时钟控制（RCC）子系统

表90. CLK_SM_0

SM代码	CLK_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对时钟和复位系统的配置寄存器应用该方法（参见RCC寄存器映射）。关于实现该方法的详细信息，可在 第 3.6.5 节 中找到
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表91. CLK_SM_1

SM代码	CLK_SM_1
说明	时钟安全系统(CSS)
所有权	ST
具体实现	时钟安全系统（CSS）检测HSE时钟的丢失并执行相应的恢复操作，例如： <ul style="list-style-type: none"> – 关闭HSE – 通过HIS通信 – 相关NMI生成
错误报告	NMI
故障检测时间	取决于具体实现（时钟频率值）
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	启动稳定后必须在时钟中断寄存器（RCC_CIR）上使能CSS保护
周期性	连续
诊断测试	CLK_SM_0：配置寄存器的定期回读
多重故障保护	CPU_SM_5：外部看门狗
建议和已知限制	建议仔细阅读关于NMI生成的参考手册说明，以便通过应用软件功能正确管理故障情况。

表92. CLK_SM_2

SM代码	CLK_SM_2
说明	独立看门狗
所有权	ST
具体实现	独立看门狗IWDG能够检测内部主要MCU时钟（低频）的故障。
错误报告	复位信号生成
故障检测时间	取决于实现（看门狗超时间隔）
已解决故障模型	永久
取决于MCU配置	无
初始化	IWDG激活。建议在选项字节设置中使用“硬件看门狗”（在复位后自动使能IWDG）。
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_1：应用软件中的控制流监控
建议和已知限制	如果使用IWDG窗口选项，终端用户必须考虑应用软件执行中可能存在的容差，以避免伪错误报告（影响系统可用性）。

表93. CLK_SM_3

SM代码	CLK_SM_3
说明	内部时钟交叉测量
所有权	终端用户
具体实现	使用TIM14功能实现该方法，TIM14功能将由32 KHz RTC时钟或外部时钟源（如果有）馈给。将TIM14计数器进度与另一个计数器（通过内部时钟馈给）进行比较。因此，可以检测振荡器频率的异常值。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	定期
诊断测试	不需要
多重故障保护	CPU_SM_1：应用软件中的控制流监控 CPU_SM_5：外部看门狗
建议和已知限制	处理瞬时故障的效率可忽略不计。在永久时钟相关故障模式覆盖率下，其效率仅为中等水平。

3.6.20 独立看门狗（IWDG），系统窗口看门狗（WWDG）

表94. WDG_SM_0

SM代码	WDG_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对WDG或WDG配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表95. WDG_SM_1

SM代码	WDG_SM_1
说明	启动时的看门狗软件测试
所有权	终端用户
具体实现	该安全机制确保所使用内部看门狗的正常工作。启动时，软件测试为看门狗设定需要的过期超时，在SRAM中保存特定非平凡代码并等待复位信号。在看门狗复位后，软件获悉看门狗已正确触发，因此不再执行该程序。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	启动（参见下文）
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	在典型的终端用户应用中，只能在启动时以及维护或离线期间执行该测试。这与IEC61508中的“验证测试”概念有关，因此在操作时间内其诊断覆盖率贡献不能计算在内。

3.6.21 调试

表96. DBG_SM_0

SM代码	DBG_SM_0
说明	独立看门狗
所有权	ST
具体实现	硬件随机故障导致的调试意外激活将造成CPU操作的巨大干扰，引起独立看门狗或其他系统看门狗WWGDG或外部看门狗的干预。
错误报告	复位信号生成
故障检测时间	取决于实现（看门狗超时间隔）
已解决故障模型	永久
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_1：应用软件中的控制流监控
建议和已知限制	无

3.6.22 循环冗余校验模块（CRC）

表97. CRC_SM_0

SM代码	CRC_SM_0
说明	CRC自我覆盖
所有权	ST
具体实现	该模块中实现的CRC算法（CRC-32以太网多项式：0x4C11DB7）提供出色的消息错误检测功能。因此，通过任何使用模块重新计算预期签名的操作，都可以容易地检测到影响CRC计算的永久和瞬时故障。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	无
初始化	取决于实现
周期性	连续
诊断测试	不需要
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	无

3.6.23 系统配置控制器（SYSCFG）

表98. SYSCFG_SM_0

SM代码	SYSCFG_SM_0
说明	配置寄存器的定期回读
所有权	终端用户
具体实现	必须对系统配置控制器配置寄存器应用该方法。 强烈建议使用该方法保护与硬件诊断激活和错误报告链相关功能有关的寄存器。 关于实现该方法的详细信息，可在第 3.6.5 节中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	该方法主要与其他MCU外设所需的多个其他“配置寄存器回读”存在重叠。记录在这里是出于完整性的考虑。

表99. DIAG_SM_0

SM代码	DIAG_SM_0
说明	硬件诊断配置寄存器的定期回读
所有权	终端用户
具体实现	在STM32L0系列中，有多个基于硬件的高效安全机制（例如，闪存上的ECC）可供使用。应对与诊断措施操作（包括错误报告）相关的任何配置寄存器应用该方法。因此，终端用户应区别对待与以下内容有关的配置寄存器： <ul style="list-style-type: none"> – 硬件诊断使能 – 中断/NMI使能（如果用于诊断错误管理）
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

3.6.24 灵活的静态存储控制器（FSMC）

表100. FSMC_SM_0

SM代码	FSMC_SM_0
说明	应用软件中的控制流监控
所有权	终端用户
具体实现	如果使用FSMC连接外部存储器，而该外部存储器包含要由CPU执行的软件代码，则影响FSMC存储器控制器的永久和瞬时故障能够干扰CPU的存取操作，导致错误的数据或指令提取。当此类故障干扰预期的应用软件流时，连接到系统看门狗的控制流机制能够检测到此类故障。 该方法的实现与CPU_SM_1报告的相同，参见CPU_SM_1了解详情。
错误报告	取决于实现
故障检测时间	取决于实现。看门狗超时时间间隔为上限。
已解决故障模型	永久和瞬时
取决于MCU配置	FSMC接口只对某些产品编号可用
初始化	取决于实现
周期性	连续
诊断测试	N/A
多重故障保护	CPU_SM_0：核心定期自检软件
建议和已知限制	仅当使用FSMC外部存储器保存可执行程序时，才能使用该机制。

表101. FSMC_SM_1

SM代码	FSMC_SM_1
说明	连接到FSMC的外部存储器上的信息冗余
所有权	终端用户
具体实现	如果使用FSMC接口连接保存有安全相关数据的外部存储器，已保存数据的信息冗余技术能够处理影响FSMC接口的故障。可能的技术为： 使用安全相关数据的冗余副本并在使用前执行一致性检查。 在使用前，对数据进行校验计算。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	FSMC接口只对某些产品编号可用
初始化	取决于实现
周期性	按需
诊断测试	不需要

表101. FSMC_SM_1 (续)

多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	仅当使用FSMC外部存储器保存安全相关数据时, 才能使用该机制。 该安全机制与在系统层面实现的信息冗余技术 (用于处理连接到FSMC端口的物理设备的故障) 之间可能存在重叠。

表102. FSMC_SM_2

SM代码	FSMC_SM_2
说明	FSMC配置寄存器的定期回读
所有权	终端用户
具体实现	必须对FSMC配置寄存器应用该方法。 关于实现该方法的详细信息, 可在第 3.6.5 节中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	FSMC接口只对某些产品编号可用
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表103. FSMC_SM_3

SM代码	FSMC_SM_3
说明	FSMC模块的NAND接口上的ECC引擎
所有权	ST
具体实现	FMQAND卡控制器包括两个纠错码计算硬件模块, 每个存储区域各有一个。这些模块可在软件处理 ECC 时减少主机 CPU 工作负载。 ECC机制保护连接到NAND端口的的外部存储器上的数据完整性。
错误报告	参见功能文档
故障检测时间	在存储器读取期间检查ECC位。
已解决故障模型	永久和瞬时
取决于MCU配置	FSMC接口只对某些产品编号可用
初始化	无
周期性	连续
诊断测试	不需要

表103. FSMC_SM_3 (续)

多重故障保护	FSMC_SM_2: FSMC配置寄存器的定期回读
建议和已知限制	在检测影响FSMC接口的硬件随机故障方面, 该方法的效力可忽略不计。由于在STM32L0 MCU之外进行存储器寻址, 它可以是终端用户安全理念的一部分。

3.6.25 真随机数发生器 (RNG)

表104. RNG_SM_0

SM代码	RNG_SM_0
说明	RNG配置寄存器RNG_CR的定期回读
所有权	终端用户
具体实现	必须对RNG配置寄存器应用该方法。 关于实现该方法的详细信息, 可在 第 3.6.5 节 中找到
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	RNG模块只对特定产品编号可用
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表105. RNG_SM_1

SM代码	RNG_SM_1
说明	RNG模块熵在线测试
所有权	ST和终端用户
具体实现	RNG模块包含模拟源熵的内部诊断, 可用于检测模块本身的故障。此外, 可以使用生成的随机数与上一个随机数之间的差异测试 (按照FIPS PUB 140-2的要求)。 实现: – 检查RNG错误条件 – 检查生成的随机数与上一个随机数之间的差异
错误报告	RNG状态寄存器 (RNG_SR) 中的CEIS、SEIS错误位 导致FIPS PUB 140-2测试失败的应用软件错误
故障检测时间	取决于实现
已解决故障模型	永久和瞬时
取决于MCU配置	RNG模块只对特定产品编号可用
初始化	取决于实现

表105. RNG_SM_1 (续)

周期性	连续
诊断测试	N/A
多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	-

3.6.26 高级加密标准硬件加速器 (AES)

表106. AES_SM_0

SM代码	AES_SM_0
说明	AES配置寄存器的定期回读
所有权	终端用户
具体实现	必须对AES配置寄存器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	AES模块只对特定产品编号可用
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表107. AES_SM_1

SM代码	AES_SM_1
说明	加密/解密辅助检测
所有权	ST
具体实现	AES模块执行的加密和解密操作由多种数据操作和检查构成，根据所选链算法，有不同的复杂度级别。大部分影响AES模块的硬件随机故障会导致算法违规/错误。导致接收器侧的解码错误。
错误报告	可能发生多种错误状态，请参见功能文档
故障检测时间	取决于外设配置。
已解决故障模型	永久和瞬时
取决于MCU配置	AES模块只对特定产品编号可用
初始化	取决于实现
周期性	连续

表107. AES_SM_1 (续)

诊断测试	N/A
多重故障保护	AES_SM_2: 消息的信息冗余技术
建议和已知限制	-

表108. AES_SM_2

SM代码	AES_SM_2
说明	消息的信息冗余技术, 包括端到端安全
所有权	终端用户
具体实现	该方法旨在保护外设与其外部对等体之间的通信。在AES局部安全理念中, 该方法用于处理加密/解密功能检测不到的故障。 参见CAN_SM_2说明获取详细信息。
错误报告	请参考CAN_SM_2
故障检测时间	请参考CAN_SM_2
已解决故障模型	请参考CAN_SM_2
取决于MCU配置	AES模块只对特定产品编号可用
初始化	请参考CAN_SM_2
周期性	请参考CAN_SM_2
诊断测试	请参考CAN_SM_2
多重故障保护	请参考CAN_SM_2
建议和已知限制	<i>注:</i> 假定通信对象具有执行所述检查的同等能力。 参见CAN_SM_2获取更多说明。

注: 本手册中不分析硬件随机故障的潜在安全功能违规后果。

3.6.27 防火墙 (FW)

表109. FWR_SM_0

SM代码	FWR_SM_0
说明	防火墙配置寄存器的定期回读
所有权	终端用户
具体实现	必须对防火墙配置寄存器 (同样未被终端用户应用软件使用) 应用该方法。 关于实现该方法的详细信息, 可在第 3.6.5 节中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0

表109. FWR_SM_0 (续)

周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

3.6.28 液晶显示控制器 (LCD)

表110. LCD_SM_0

SM代码	LCD_SM_0
说明	LCD配置寄存器和缓冲存储器的定期回读。
所有权	终端用户
具体实现	必须对LCD配置寄存器和缓冲存储器应用该方法。 关于实现该方法的详细信息，可在 第 3.6.5 节 中找到。
错误报告	请参考NVIC_SM_0
故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0

表111. LCD_SM_1

SM代码	LCD_SM_1
说明	通过ADC通道执行LCD采集
所有权	终端用户
具体实现	通过比较ADC读数和预期值检查是否正确生成了LCD驱动信号。
错误报告	取决于实现
故障检测时间	取决于实现
已解决故障模型	永久
取决于MCU配置	无
初始化	无
周期性	定期
诊断测试	不需要

表111. LCD_SM_1 (续)

多重故障保护	CPU_SM_0: 核心定期自检软件
建议和已知限制	该方法主要用于检测影响模拟部件的永久故障, 因此可以接受定期执行。可达到的诊断覆盖率取决于检查的LCD信号的数量。

注: 上述安全机制处理STM32MCU中包含的LCD接口。由于该安全机制不处理在LCD上正确生成图像的实际能力, 当此类功能被认为与安全相关时, 将警告终端用户评估是否采用充分的系统层面措施。

3.6.29 禁用并定期交叉检查未使用外设的意外激活

本节描述针对安全应用未使用的外设或未完全使用的外设的安全机制。

表112. FFI_SM_0

SM代码	FFI_SM_0
说明	禁用未使用的外设
所有权	终端用户
具体实现	该方法有助于降低软件应用未使用的外设导致交叉干扰的可能性, 以防止硬件故障导致意外激活。 在系统启动后, 应用软件必须通过该程序禁用所有未使用的外设: – 使能AHB和APB外设复位寄存器上的复位标志 – 禁用AHB和APB外设时钟使能寄存器上的时钟分配
错误报告	NA
故障检测时间	NA
已解决故障模型	NA
取决于MCU配置	无
初始化	NA
周期性	启动
诊断测试	不需要
多重故障保护	FFI_SM_1: 干扰避免寄存器的定期回读
建议和已知限制	无

表113. FFI_SM_1

SM代码	FFI_SM_1
说明	干扰避免寄存器的定期回读
所有权	终端用户
具体实现	该方法有助于降低外设间发生交叉干扰的可能性, 外设可能在相同输入/输出引脚上发生冲突, 包括未使用的外设等。必须对以下寄存器应用该诊断措施: – 时钟使能和禁用寄存器 – 复用功能编程寄存器 关于实现该方法的详细信息, 可在第 3.6.5 节中找到。
错误报告	请参考NVIC_SM_0

表113. FFI_SM_1 (续)

故障检测时间	请参考NVIC_SM_0
已解决故障模型	请参考NVIC_SM_0
取决于MCU配置	请参考NVIC_SM_0
初始化	请参考NVIC_SM_0
周期性	请参考NVIC_SM_0
诊断测试	请参考NVIC_SM_0
多重故障保护	请参考NVIC_SM_0
建议和已知限制	请参考NVIC_SM_0



3.7 使用条件

表 114 对第 3.6 节中报告的安全机制建议进行了总结。要应用于 STM32L0xx MCU 的使用条件以安全机制要求的形式记录。例外情况是为了正确解决特定的故障模式而通过 FMEA 分析引入的一些使用条件。这些使用条件记录在表 114 的末尾。

“等级”栏记录了如何在分析过程中考虑了相关安全机制，符号的含义如下：

- M = 该安全机制总是在正常操作期间工作 – 没有任何终端用户活动能够停用它。
- ++ = 强烈建议作为常规做法且本安全手册中考虑用于计算安全指标，以便在单个 MCU 上达到 SIL2。
- + = 建议作为额外的安全措施，但本安全手册中未考虑用于计算安全指标。如果它与功能要求相矛盾或与另一种被标记为“++”的安全机制发生重叠，STM32L0 系列的用户可以跳过该实现。
- o = 可选，不需要或与特定 MCU 配置相关

永久和瞬时列中的“X”标记表示相关安全机制对此类故障模式有效。

表114. 安全机制列表

STM32L0系列的功能	诊断	说明	等级	永久	瞬时
Arm® Cortex®-M3 CPU	CPU_SM_0	解决Arm®Cortex®-M3CPU核心中永久故障的定期软件测试	++	X	-
	CPU_SM_1	应用软件中的控制流监控	++	X	X
	CPU_SM_2	应用软件中的双重计算	++	-	X
	CPU_SM_3	Arm® Cortex®-M3 HardFault异常	M	X	X
	CPU_SM_4	应用软件的堆栈加固	+	X	X
	CPU_SM_5	外部看门狗	+(1)	X	X
	CPU_SM_6	独立看门狗	++(1)	X	X
	CPU_SM_7	MPU - 存储器保护单元	++(2)	X	X
	MPU_SM_0	MPU配置寄存器的定期回读	++(2)	X	X
嵌入式 Flash	FLASH_SM_0	闪存的定期软件测试	++	X	-
	FLASH_SM_1	应用软件中的控制流监控	++	X	X
	FLASH_SM_2	Arm® Cortex®-M3 HardFault异常	M	X	X
	FLASH_SM_3	选项字节写保护	M	-	-
	FLASH_SM_4	静态数据封装	+	X	X
	FLASH_SM_5	具有负载验证的选项字节冗余	M	X	X
	FLASH_SM_6	未使用闪存区填充码	+	-	-
	FLASH_SM_8	读/写/专有代码保护	+	-	-
内部 SRAM	RAM_SM_0	SRAM的定期软件测试	++	X	-
	RAM_SM_2	应用软件的堆栈加固	+	X	X
	RAM_SM_3	应用软件中系统变量的信息冗余	++	X	X
	RAM_SM_4	应用软件中的控制流监控	o(3)	X	X
	RAM_SM_5	RAM中应用软件的定期完整性测试	o(3)	X	X

表114. 安全机制列表（续）

STM32L0系列的功能	诊断	说明	等级	永久	瞬时
系统架构	BUS_SM_0	互连的定期软件测试	++	X	-
	BUS_SM_1	片内数据交换中的信息冗余	++	X	X
系统EEPROM	EEP_SM_0	信息冗余	++	X	X
	EEP_SM_1	写操作后的软件回读	+	X	X
EXTI控制器	NVIC_SM_0	配置寄存器的定期回读	++	X	X
	NVIC_SM_1	通过应用软件执行预期和意外中断检查	++	X	X
DMA	DMA_SM_0	配置寄存器的定期回读	++	X	X
	DMA_SM_1	通过DMA传输的数据包的信息冗余	++	X	X
	DMA_SM_2	通过DMA传输的数据包的信息冗余，包括发送者和接收者标识符	++	X	X
	DMA_SM_3	DMA的定期软件测试	++	X	-
	DMA_SM_4	DMA事务感知	++	X	X
USART, LPUART	UART_SM_0	配置寄存器的定期回读	++	X	X
	UART_SM_1	协议错误信号	++	X	X
	UART_SM_2	消息的信息冗余技术	++	X	X
	UART_SM_3	消息的信息冗余技术，包括端到端安全	++	X	X
I2C:	IIC_SM_0	配置寄存器的定期回读	++	X	X
	IIC_SM_1	协议错误信号	++	X	X
	IIC_SM_2	消息的信息冗余技术	++	X	X
	IIC_SM_3	CRC数据包层面	+	X	X
	IIC_SM_4	消息的信息冗余技术，包括端到端安全	+	X	X
SPI	SPI_SM_0	配置寄存器的定期回读	++	X	X
	SPI_SM_1	协议错误信号	++	X	X
	SPI_SM_2	消息的信息冗余技术	++	X	X
	SPI_SM_3	CRC数据包层面	+	X	X
	SPI_SM_4	消息的信息冗余技术，包括端到端安全	+	X	X
USB:	USB_SM_0	配置寄存器的定期回读	++	X	X
	USB_SM_1	协议错误信号	++	X	X
	USB_SM_2	消息的信息冗余技术	++	X	X
	USB_SM_3	消息的信息冗余技术，包括端到端安全	+	X	X
TSC	TSC_SM_0	配置寄存器的定期回读	++	X	X
	TSC_SM_1	通过应用软件进行多重采集	++	-	X
	TSC_SM_2	TSC采集永久故障的应用层面检测	+	X	-

表114. 安全机制列表（续）

STM32L0系列的功能	诊断	说明	等级	永久	瞬时
ADC	ADC_SM_0	配置寄存器的定期回读	++	X	X
	ADC_SM_1	通过应用软件进行多重采集	++	-	X
	ADC_SM_2	通过应用软件进行范围检查	++	X	X
	ADC_SM_3	ADC的定期软件测试	++	X	-
	ADC_SM_4	ADC输入的1002方案	+	X	X
DAC	DAC_SM_0	配置寄存器的定期回读	++	X	X
	DAC_SM_1	ADC通道上的DAC输出环回	++	X	X
COMP	COMP_SM_0	配置寄存器的定期回读	++	X	X
	COMP_SM_1	比较器的1002方案	++	X	X
	COMP_SM_2	输入的真实性检查	+	X	-
	COMP_SM_3	通过应用软件进行多重采集	+	-	X
	COMP_SM_4	比较器锁定机制	+	-	-
基本定时器TIM 6/7	GTIM_SM_0	配置寄存器的定期回读	++	X	X
	GTIM_SM_1	计数定时器的1002	++	X	X
高级、通用和低功耗定时器TIM2/3/21/22/LPTIM1	ATIM_SM_0	配置寄存器的定期回读	++	X	X
	ATIM_SM_1	计数定时器的1002	++	X	X
	ATIM_SM_2	输入捕获定时器的1002	++	X	X
	ATIM_SM_3	PWM输出的环回方案	++	X	X
	ATIM_SM_4	定时器的锁定位保护	+	-	-
CRC	CRC_SM_0	CRC自我覆盖	++	X	X
GPIO	GPIO_SM_0	配置寄存器的定期回读	++	X	X
	GPIO_SM_1	输入GPIO线的1002	++	X	X
	GPIO_SM_2	输出GPIO线的环回模式	++	X	X
	GPIO_SM_3	GPIO 端口配置锁定寄存器	+	-	-
RTC	RTC_SM_0	配置寄存器的定期回读	++	X	X
	RTC_SM_1	运行RTC的应用检查	++	X	X
	RTC_SM_2	备份寄存器的信息冗余	+	X	X
	RTC_SM_3	用于检测时间戳或事件捕获中的故障的应用层面措施	0	X	X
电源控制	VSUP_SM_0	配置寄存器的定期回读	++	X	X
	VSUP_SM_1	电源电压监测	++	X	-
	VSUP_SM_2	独立看门狗	++	X	-
	VSUP_SM_3	内部温度传感器检查	0	-	-
	VSUP_SM_4	外设电压监测（PVM）	+	X	-

表114. 安全机制列表（续）

STM32L0系列的功能	诊断	说明	等级	永久	瞬时
时钟和复位	CLK_SM_0	配置寄存器的定期回读	++	X	X
	CLK_SM_1	CSS时钟安全系统	++	X	-
	CLK_SM_2	独立看门狗	++	X	-
	CLK_SM_3	内部时钟交叉测量	+	X	-
IWDG/WWDG	WDG_SM_0	配置寄存器的定期回读	++	X	X
	WDG_SM_1	启动时的看门狗软件测试	o	X	-
调试	DBG_SM_0	独立看门狗	++	X	X
系统或外设控制	LOCK_SM_0	配置选项的锁定机制	+	-	-
	SYSCFG_SM_0	配置寄存器的定期回读	++	X	X
	DIAG_SM_0	硬件诊断配置寄存器的定期回读	++	X	X
灵活的静态存储控制器 (FSMC)	FSMC_SM_0	应用软件中的控制流监控	++ ⁽⁴⁾	X	X
	FSMC_SM_1	连接到FSMC的外部存储器上的信息冗余	++ ⁽⁴⁾	X	X
	FSMC_SM_2	FSMC配置寄存器的定期回读。	++	X	X
	FSMC_SM_3	FSMC模块的NAND接口上的ECC引擎	o	X	X
RNG	RNG_SM_0	RNG配置寄存器RNG_CR的定期回读。	++	X	X
	RNG_SM_1	RNG模块熵在线测试	++	X	-
AES	AES_SM_0	AES配置寄存器的定期回读	++	X	X
	AES_SM_1	加密/解密辅助检测	++	X	X
	AES_SM_2	消息的信息冗余技术，包括端到端安全	++	X	X
防火墙 (FW)	FWR_SM_0	防火墙配置寄存器的定期回读	o	X	X
LCD	LCD_SM_0	LCD配置寄存器和缓冲存储器的定期回读。	++	X	X
	LCD_SM_1	通过ADC通道执行LCD采集	++	X	-
部件隔离（无干扰）	FFI_SM_0	禁用未使用的外设	++	-	-
	FFI_SM_1	干扰避免寄存器的定期回读	++	-	-
Arm® Cortex®-M3 CPU	CoU_1	Arm®Cortex®-M3CPU的复位状态必须与系统层面的有效安全状态兼容	++	-	-
调试	CoU_2	不得在安全功能实现中使用STM32L0系列调试功能	++	-	-
Arm® Cortex®-M3 / 电源系统	CoU_3	不得在安全功能实现中使用低功耗模式状态	++	-	-
STM32L0系列外设	CoU_4	终端用户必须为安全功能实现中使用的每个STM32外设实现要求的安全机制/CoU组合。	++	X	X
Flash子系统	CoU_5	在闪存区批量擦除和重新编程期间，STM32MCU不能执行任何安全功能。	++	-	-
Flash子系统	CoU_6	通过双闪存系统执行的现场应用软件实时更新必须包括如FLASH_SM_0等方法执行的代码/数据完整性检查	++	X	X

表114. 安全机制列表（续）

STM32L0系列的功能	诊断	说明	等级	永久	瞬时
CPU子系统	CoU_7	如果实现多重安全功能，用于保证其相互独立性的方法必须包含MPU的使用。	++	-	-
CRS	CoU_8	不得在安全功能实现中使用CRS功能	++	-	-

1. 为了在单个MCU上达到SIL2，将CPU_SM_5评级为“+”。总之，针对系统层面安全状态的特殊定义，可能必需将CPU_SM_5评级为“++”；这种情况下，可以将CPU_SM_6评级为“+”。更多详细信息，请参见表 11：CPU_SM_6的“建议”一行。
2. 如果没有实现多重安全功能，可考虑评级为“+”。
3. 如果在RAM上执行应用软件，必需考虑评级为“++”。
4. 根据连接到FSMC的外部存储器的预定用途，可考虑评级为“o”。

上述安全机制或使用条件根据其性质设计有不同抽象级别：实现的安全机制越是独立于应用，它在各种终端用户应用上的可能用途越广泛。

安全分析突出了MCU中的两个主要部分：

- 系统关键型MCU模块。从安全角度来看，每个终端用户应用都会受到这些模块上的故障的影响。由于每个终端用户应用都使用这些模块，相关方法或安全机制的设计大体上独立于应用。STM32L0系列微控制器的系统关键型模块为：CPU、复位、功耗、时钟、总线矩阵与互联、闪存与RAM（包括其接口）。
- 外设模块。此类模块可以不被终端用户应用使用，或者可以用于非安全相关任务。因此，相关安全方法主要是在应用层面作为应用软件解决方案或架构解决方案来实现。

4 安全结果

本节报告STM32L0系列MCU安全分析结果。该报告遵循IEC61508和ST方法流程，与硬件随机和从属故障相关。

4.1 随机硬件故障安全结果

本安全手册中报告的STM32L0系列器件随机硬件故障分析按照ST的半导体器件安全分析方法流程执行（依据IEC61508）。通过三个因素保证获得的结果的准确性：

- ST方法流程严格符合IEC61508的要求和指示
- 关于微控制器设计的详细可靠信息的分析
- 安全指标验证的先进故障注入法和工具的使用

因此，STM32L0系列安全分析能够探索MCU故障模式的完整详尽列表，并对其中的每一种分别采取充分的缓解措施（安全机制）。相关FMEA文档中提供了STM32L0系列故障模式的完整列表。STM32L0系列FMEA文档可按需提供，请咨询您当地的ST销售联系人。

总之，通过采用 [第 3.7 节：使用条件](#) 中报告的安全机制和使用条件，可以达到 [表 115](#) 中总结的完整性等级。

表115. 整体可达到的安全完整性等级

使用的MCU	安全架构	目标	安全分析结果
1	1001/1001D	SIL2 LD	可达到
		SIL2 HD/CM	可达到并可能存在性能影响 ⁽¹⁾
2	1002	SIL3 LD	可达到
		SIL3 HD/CM	可达到并可能存在性能影响

1. 请注意，与上文报告的某些目标达成相关的潜在性能影响主要与基于软件的定期诊断的执行需求有关（参见安全机制说明了解详情）。因此，影响与系统层面PST的“积极”程度紧密相关（参见 [第 3.3.1 节：假定安全要求](#)）。

本节没有记录得到的相对安全指标（DC和SFF）和绝对安全指标（PFH和PFD），但FMEDA快照中有，原因如下：

- STM32L0系列产品编号的数量很大，
- 声明非安全相关未使用外设的可能性，以及
- 使能或不使能不同的可用安全机制的可能性。

FMEDA快照是一种静态文档，它报告给定安全机制组合和给定产品编号的以不同细节水平计算的安全指标（在微控制器层面并针对微控制器基础功能）。如需FMEDA计算表，请联系当地的意法半导体销售代表，获取特定MCU目标产品编号的预期交付日期的相关信息。

注：安全指标计算仅限于STM32L0系列，因此不包括WDTe、PEv和VMOne（[第 3.2 节](#)中对它们进行了描述）

4.1.1 安全分析结果自定义

为STM32L0系列器件执行的并包含在本安全手册中的安全分析考虑了所有与安全相关的微控制器模块，因此能够干预安全功能，无一例外。这与通用微控制器分析期间要遵循的保守方法相一致，目的是实现相对于最终应用的独立性。这意味着按照IEC61508-4第3.6.8款，没有任何微控制器模块被声明为“安全”，因此SFF计算中包含所有微控制器模块。

在实际的终端用户应用中，并非所有STM32L0系列部件或模块都被用于安全功能的实现。这可能发生在其他两种可能的情况下：

- 部件根本未被使用（禁用）
- 部件用于实现非安全相关（例如，驱动电子板上“上电”LED指示灯的GPIO线）的功能。

要求对这些未使用部件实现安全机制可导致过度杀伤。因此，可以自定义安全分析结果。

终端用户可以将所选的处于以下条件下的STM32L0系列部件定义为“非安全相关”（由终端用户负责）：

- 收集这些部件在安全功能实现中不发挥作用的理由和证据。
- 收集这些部件在正常操作期间因最终系统设计决策导致其不干预安全功能的理由和证据。
- 满足下列缓解MCU内部干扰的一般条件（表 116）。

因此，终端用户允许“非安全相关”部件执行以下操作：

- 从FMEDA的指标计算中舍弃部件贡献；
- 不实施表 114中列出的相关安全机制。

就SFF计算而言，该程序等同于按照IEC61508-4第3.6.13/14款（已舍弃模块的任何故障定义）声明“无部件/无影响”。

4.1.2 免受干扰（FFI）的一般要求

通过专门的分析列出了在发生内部故障时为缓解STM32L0系列内部模块之间的潜在干扰需要遵循的一般要求清单（免受干扰，FFI）。这些预防措施是STM32L0系列安全理念的组成部分，当终端用户如第 4.1.1 节所述将多个微控制器模块声明为“非安全相关”时，这些预防措施可以发挥重要作用。

对终端用户的要求是实现表 116中列出的安全机制（实现详情可以在硬件和软件诊断说明中找到），不考虑关于它们对安全指标计算所做贡献的任何评估。

表116. FFI的一般要求列表

诊断	说明
FFI_SM_0	禁用未使用的外设
FFI_SM_1	干扰避免寄存器的定期回读
BUS_SM_0	互连的定期软件测试
NVIC_SM_0	配置寄存器的定期回读
NVIC_SM_1	通过应用软件执行预期和意外中断检查
DMA_SM_0	配置寄存器的定期回读
DMA_SM_2	通过DMA传输的数据包的信息冗余，包括发送者和接收者标识符 ⁽¹⁾

表116. FFI的一般要求列表（续）

诊断	说明
DMA_SM_4	DMA事务感知 ⁽¹⁾
GPIO_SM_0	配置寄存器的定期回读

1. 仅在实际使用了DMA时实现

4.1.3 关于多故障情景的说明

原则上，IEC61508需要分析多故障情景，因此仅限于一次一种故障是不可接受的。STM32L0系列的安全分析也相应地考虑了多故障情景。此外，根据ISO26262（集成电路安全分析的参考和先进标准规范）的精神，该分析调查了会“禁用”规定的安全机制的故障，以便为此类情况提供个性化的缓解措施。第 3.6 节表格中的“多故障保护”字段描述了正确管理多故障情景所需的相关安全机制，包括针对安全机制禁用的缓解措施。

强烈建议在安全机制中包括此类缓解措施的实现。对于必须考虑错误累积问题的长期操作系统，这一点尤为重要。

4.2 从属故障分析

微控制器的从属故障分析十分重要。从属故障的主要子类是共因故障（CCF）。其分析按照IEC 61508:2附录E进行判定，该附录列出了为允许对具有一个共用半导体衬底的IC使用片上冗余要进行验证的设计要求。但是，附录E.1和E.2适用于HFT=1，而附录E.3必须应用于每个片上冗余，在同一硅晶上实现的诊断也是如此。

由于STM32L0系列器件无片上冗余，因此无需通过BetaIC算法实现CCF量化。请注意，对于1oo2安全架构实现，要求终端用户评估参数βD，即PFH计算中使用的两个通道间共因的度量。

STM32L0系列器件的架构和结构可能是从属故障的潜在源头。下面几节将对这些情况进行分析。所述安全机制的详细描述见第 3.6 节：硬件和软件诊断说明节。

4.2.1 电源

电源是从属故障的潜在源头，因为功率的任何变化都可能影响许多部件，导致非独立故障。以下安全机制可解决和缓解这些从属故障：

- VSUP_SM_1：电源电压异常值的检测
- VSUP_SM_2：独立看门狗拥有和MCU数字核心的不同电源，这种多样性有助于缓解与主电源变化相关的从属故障。

因此，强烈建议采用此类安全机制，尽管它们对达到安全完整性等级所需的安全指标贡献不大。参见第 3.6.19 节：复位和时钟控制（RCC）子系统获取详细的安全机制说明。

4.2.2 时钟

系统时钟是从属故障的潜在源头，因为时钟特性（频率、抖动）的变化可能影响许多部件，导致非独立故障。以下安全机制可解决和缓解这些从属故障：

- CLK_SM_1：时钟安全系统能够检测系统时钟的改变（停止）并激活适当的恢复操作。
- CLK_SM_2：独立看门狗具有专用时钟源。系统时钟的频率改变会通过触发应用软件上的例程引起看门狗窗口违规，导致看门狗执行MCU复位。



因此，强烈建议采用此类安全机制，尽管它们对达到安全完整性等级所需的安全指标贡献不大。参见 [第 3.6.20 节：独立看门狗 \(IWDG\)](#)，[系统窗口看门狗 \(WWDG\)](#) 获取详细的安全机制说明。

4.2.3 DMA

DMA是主要由所有外设操作的数据传输中一种被广泛共享的资源。DMA故障可能干扰系统外设或应用软件的行为，导致非独立故障。解决此类从属故障的安全机制如下：

- DMA_SM_0
- DMA_SM_1
- DMA_SM_2。

因此，强烈建议采用此类安全机制。值得注意的是，如果不使用DMA进行数据传输，则只需实现DMA_SM_0。参见 [第 3.6.6 节：直接存储器访问控制器 \(DMA\)](#) 获取详细的安全机制说明。

4.2.4 内部温度

内部温度异常上升是从属故障的潜在源头，因为它可能影响许多MCU部件，从而导致非独立故障。用于缓解此潜在影响的安全机制如下：

- VSUP_SM_3：内部温度读取和检查使用户能够快速检测潜在风险条件，防止其导致一系列内部故障。参见 [第 节：电源控制](#) 获取详细的安全机制说明。

5 证据列表

安全案例数据库保存了为获得本安全手册中报告的结果和结论而执行的安全分析的所有相关信息。

安全案例数据库具体包含以下内容：

- 具有所有安全分析相关文档完整列表的安全案例
- ST内部FMEDA工具数据库用于计算安全指标，包括估计值和实测值
- 安全报告是指详细描述在STM32L0系列器件上按照IEC61508逐条执行的安全分析的文档
- ST内部故障注入活动数据库包含工具配置和设置、故障注入日志和结果

由于存在ST机密信息，上述内容未公开提供，只提供给可能的合格验证机构用于审计和检查。这符合IEC61508:2第7.4.9.7款注释2的说明。

附录A 其他安全标准的变更影响分析

本安全手册中描述的安全分析按照IEC 61508安全标准执行。本附录报告不同安全标准的变更影响分析的结果。针对每个新安全标准考虑以下主题：

- 推荐硬件架构（架构类别）的差异以及如何映射到IEC 61508的安全架构。
- 安全完整性等级定义和指标计算方法的差异，以及如何按照新标准重新计算和判断STM32L0系列器件的安全性能。
- 对现在已经达到IEC61508标准的产品，如何进行重新映射和修改（如果需要的话），以使其符合新的标准。

此变更影响分析中检查的安全标准包括：

- ISO 13849-1:2006和ISO 13849-2:2010 – 机械安全 - 控制系统的安全相关部分
- IEC 62061:2012-11, 版本1.1 – 机械安全-安全相关电气\电子\可编程电子控制系统的功能安全
- IEC 61800-5-2:2007 1.0版 – 调速电气传动系统 – 第5-2部分：安全要求 - 功能
- ISO 26262:2010 – 道路车辆 - 电气或电子（EE）系统。

A.1 ISO 13849-1 / ISO 13849-2

ISO 13849-1是B1类标准。它为包括可编程电子设备、硬件和软件在内的机械控制系统安全相关部件（SRP或CS）的开发提供指南。

A.1.1 架构类别

ISO 13849的第6.2节针对不同的基础参数，系统诊断检测范围（DC），系统平均无危险故障时间（MTTFd）和共因故障预防（CCF）定义了5个类别，反映设计的控制系统（CS）和安全相关部件（SRP）所预期的容错能力和达到要求的系统安全等级（PLr）。该标准为每个类别推荐了一种满足相关要求的典型架构。

考虑到§6.2中定义的ISO 13849架构类别并以微控制器为中心，[表 117](#)为愿意开发适合安全关键型通道的逻辑运算器单元并执行指定安全功能的终端用户列出了一个总结。

相关假设如下：

1. 通过元件（SRP或CS）输入系统、信号处理单元和输出系统的系列组合实现安全功能。
2. 可将SRP或CS元件分配给一个或多个不同的类别和不同PL。
3. 安全功能完全处于终端用户应用的范围内。
4. 采用本安全手册中所述安全机制的STM32L0系列MCU作为单个合规项，可达到PLd级（相当于SIL2）的CM应用。

逻辑运算器的ISO 13849架构类别如[表 117](#)所示。

表117. ISO 13849架构分类

仅限	参考 §	总结	指定逻辑架构	框图
B	6.2.3	主要类别；发生一次故障可导致安全功能丧失。 无需DC和CCF（通常为单通道），MTTFd为低或中等。 可达到的最高PL = b	单通道架构，1oo1中一个MCU 请参见第 3 节。 合规项的MTTFd = 高	图 6
1	6.2.4	通过采用以安全关键型应用的“充分试验的组件”为基础的解决方案和“充分试验的”安全原则来实施B类要求。 微控制器未被划分为“充分试验的”组件。 无需DC和CCF（通常为单通道），MTTFd为高。 可达到的最高PL = c。	单通道架构，1oo1中一个MCU 请参见第 3 节。 合规项的MTTFd = 高	图 6
2	6.2.5	就类别1而言，架构中应包含测试设备，用于执行安全功能检查和报告其失败。总体DC为低，必须评估CCF，MTTFd的范围从低到高，最高PL = d。	单通道架构，1oo1d中一个MCU 请参见第 3 节。 合规项的MTTFd = 高 TE由终端用户负责，PL = d	图 7
3	6.2.6	就1类而言，应具有故障检测机制，并且任何单个故障都不会导致安全功能丧失。总体DC为低，必须评估CCF，MTTFd的范围从低到高，最高PL = d	双通道架构，1oo2中两个相同MCU 请参见第 3 节。 连续测试或监控 合规项的MTTFd = 高	图 8
4	6.2.7	就1类而言，应具有故障检测机制，并且任何单个故障都不会导致安全功能丧失。总体DC为高，必须评估通道的CCF，MTTFd为高，最高PL = e	双通道架构，1oo2中两个相同MCU 请参见第 3 节。 连续测试或监控 合规项的MTTFd = 高 可达到PL _e	图 8

图6. ISO 13849类的框图 B类和1类的框图 1

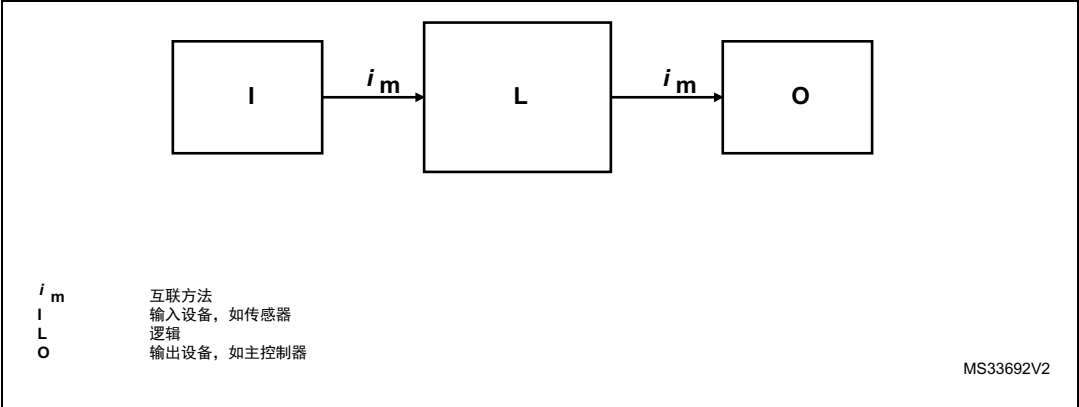


图7. ISO 13849类的框图 2

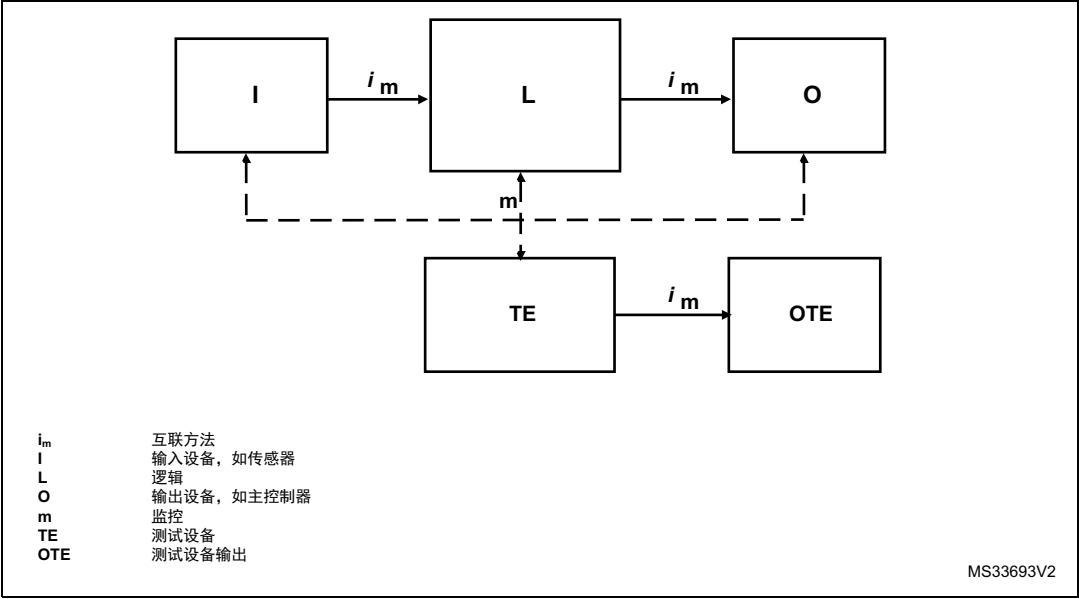
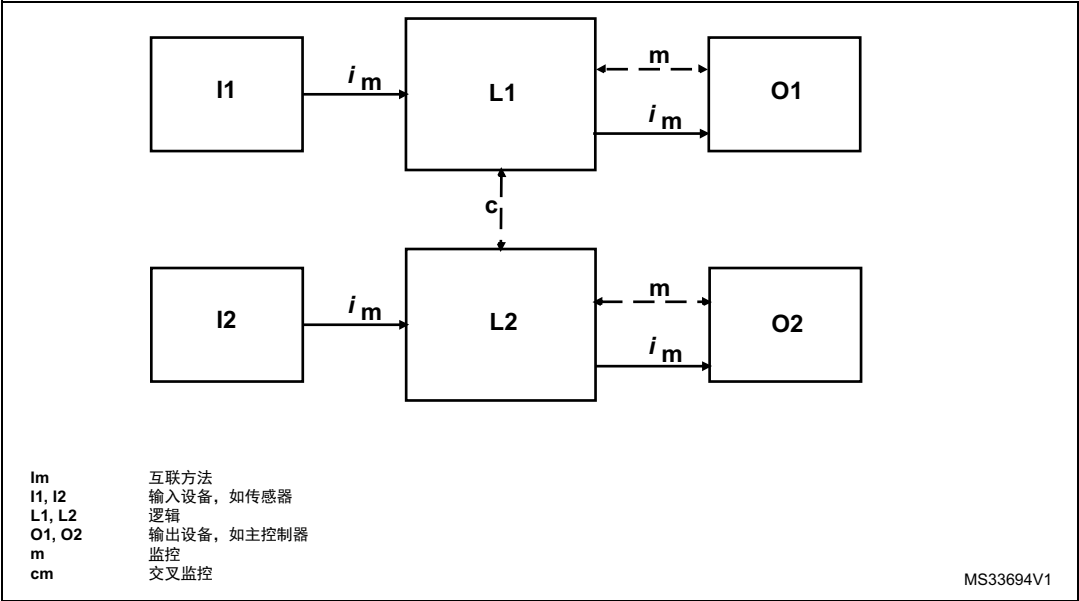


图8. ISO 13849类的框图 3类和4类的框图 4



A.1.2 安全指标计算

ISO 13849附录C提供了各种电气或电子元件的标准化MTTFd。但是，在尝试对可编程IC的MTTFd进行分类的同时，ISO 13849中的表C.3指向IC制造商的数据。因此，可以在ISO 13849的范围内重新映射本安全手册的安全分析结果，因为即使这些结果是基于IEC61508所得出它们在危险故障识别的定义方面也会越来越准确。

对于 $PFH \ll 1$ 的某个元件，我们可以假设 $MTTFd = 1 / PFH$ [年]。

从可靠性理论来看，MTTF (λ 和PFH的倒数) 是只适用于不可修复系统的指标。如今的普遍做法是对不可修复系统也使用MTBF，MTBF理解为设备首次（和唯一）故障的平均时间；这种情况下，MTBF等于MTTF。

在ISO 13849-1中，每个元件的DC具有与IEC 61508指标相同的含义；因此，可重复使用本安全手册的结果。但是，该标准以附录E中所定义公式E.1的形式定义适用于整个SRP或CS的DC_{avg}概念，其中，按照通道各子系统的MTTF计算控制系统每部分的贡献权重。在计算DC_{avg}（ISO13849-2表D.21不允许排除）的同时，该标准否认任何将故障排除在外的可能性，在本安全手册的STM32L0系列分析中采用此相同假设。

必须应用以下公式计算架构中包含2个MCU的子系统的DC_{avg}

$$DC_{avg} = \frac{\frac{DC_{MCU1}}{MTTF_{MCU1}} + \frac{DC_{MCU2}}{MTTF_{MCU2}}}{\frac{1}{MTTF_{MCU1}} + \frac{1}{MTTF_{MCU2}}}$$

对于具有相同DC和MTTF的两个相同MCU，DC_{avg} = DC。

注：对于使用两个通道实现的任何架构解决方案，需要评估可能存在的共有故障模式。ISO13849定义了IEC 61508方法的简化方法。

ISO 13849标准的表7提供了基于类别、DC_{avg}和MTTFd的SRP或CS的PL评估简化程序。值得注意的是，本安全手册中分析的每种架构解决方案得到的PFH值均可得出高MTTF值。

A.1.3 工作成果

表 118列出了ISO 13849标准要求的工作成果，以及如何将它们映射到IEC 61508合规活动的可用工作成果：

表118. ISO 13849工作成果列表

ISO 13849-1		STM32L0系列 IEC 61508文档
待提供的信息	ISO 13849-1部分-条款	
SRP或CS提供的安全功能	10 技术文档	终端用户责任
每种安全功能的特性		
安全相关部件的起始点和结束点		
环境条件		
性能等级（PL）		
选择的一个或多个类别		
与可靠性相关的参数（MTTFd、DC、CCF和任务时间）	10 技术文档	STM32L0系列 安全手册和FMEA
检测系统故障		
使用的技术；		
考虑的所有安全相关故障		
将故障排除在外的理由（参见ISO 13849-2）	10 技术文档	终端用户责任



表118. ISO 13849工作成果列表（续）

ISO 13849-1		STM32L0系列 IEC 61508文档
待提供的信息	ISO 13849-1部分-条款	
设计原理（例如考虑的故障、排除的故障）	10 技术文档	STM32L0系列 安全手册
防止合理可预见误用的措施		
对ISO 13849标准该部分的标有日期的引用（即“ISO 13849-1:2006”）；	11 使用信息	
类别（B、1、2、3或4）		
性能等级（a、b、c、d或e）		
断电的使用（参见ISO 13849-2）	G.2 系统故障控制措施	终端用户责任
控制电压击穿、电压变化、过压和欠压影响的措施		
控制或避免物理环境（例如，温度、湿度、水、振动、粉尘、腐蚀性物质、电磁干扰及其影响）影响的措施	G.2 系统故障控制措施	
包含软件的SRP或CS必须实现程序控制流监控，以发现其中的程序执行错误		
控制任何数据通信中发生的错误的影响和其他影响的措施（参见IEC 61508-2:2000第7.4.8款）		
通过自动测试执行故障检测	G.2 系统故障控制措施	STM32L0系列 安全手册
能够执行模拟或分析的计算机辅助设计工具	G.3 避免系统故障的措施	终端用户责任
模拟	-	
针对机器控制的安全相关说明	应用 J，表J.1（SW）	终端用户责任
控制架构的定义		
软件说明	应用 J，表J.1（SW）	软件用户指南 （终端用户的责任，因为终端用户负责实现基于软件的诊断）
功能块建模	应用 J，表J.1（SW）	软件要求说明 （终端用户的责任，因为终端用户负责实现基于软件的诊断）
代码中的编码注释	应用 J，表J.1（SW）	代码检查结果 （终端用户的责任，因为终端用户负责实现基于软件的诊断）
编码重新读取表		

表118. ISO 13849工作成果列表（续）

ISO 13849-1		STM32L0系列 IEC 61508文档
待提供的信息	ISO 13849-1部分-条款	
对应关系矩阵	应用 J, 表J.1 (SW)	软件模块测试说明 软件系统集成测试说明 可编程电子硬件和软件集成 测试说明 (终端用户的责任, 因为终端用户负责实现基于软件的 诊断)
测试表	应用 J, 表J.1 (SW)	软件模块测试报告 软件系统集成测试报告 可编程电子硬件和软件集成 测试报告 软件验证报告 (终端用户的责任, 因为终端用户负责实现基于软件的 诊断)

A.2 IEC 62061:2005/AMD1:2012

该标准适用于机械的安全相关电气控制系统（SRECS）的规范、设计和验证或确认。SRECS是机械的电气或电子控制系统，其故障可能导致安全性降低或丧失。SRECS实现安全相关控制功能（SRCF），以防止任何风险增加。

就安全生命周期而言，该标准的范围限于从安全要求分配到安全确认。

在更宽泛的IEC 61508:2010标准的框架内，IEC 62061是针对机械领域的专用标准。由于只是应用标准，IEC 62061对技术解决方案的要求不甚严格。此外，它侧重于安全相关控制系统的电气、电子和可编程电子部件。

请注意，IEC 62061中的§3.2.26和§3.2.27仅适用于HD或CM中的SRECS，适合机械领域。LD设备仍然受IEC 61508要求的约束。

与IEC 61508:2010的密切关系基于以下主要假设：作为子系统或子系统元件的复杂电子元件的设计必须符合IEC 61508:2010第2部分方法1H（参见 §7.4.4.2）。根据IEC 62061标准 §3.2.8中的定义，必须将微控制器作为复杂元件来考虑。

因此，本安全手册中报告的IEC 61508范围内的STM32L0系列产品的结果（参见 [第 4 节：安全结果](#)）也仍然适用于受IEC 62061管制的机械。

终端用户可以高效地采用STM32L0系列合规项，为获得SIL2或SIL3级（通过采用两个STM32L0系列MCU）机械控制设计合适的SRECS。

该标准将“子系统”（参见§3.2.5）定义为其危险故障可导致安全功能丧失的系统架构部件。

子系统可达到的完整性等级，该标准建议以HFT和SFF为基础进行分类，如 [表 119](#)所示。



表119. SIL分类与HFT

SFF	HFT		
	0	1	2
低于60%	不允许	SIL1	SIL2
介于60%和90%之间	SIL1	SIL2	SIL3
介于90%和99%之间	SIL2	SIL3	SIL3
高于99%	SIL3	SIL3	SIL3

SIL 3是此背景下对SRCF的最高要求。SIL 4不在范围之内，因为开发的最终结果是一个控制系统只对应一台机器。

对于设计者而言，必须将表格中列出的SIL值视为子系统的SILCL，其中SILCL是可以对SRECS子系统要求的最高SIL，如IEC 62061标准§3.2.24所定义。

A.2.1 架构类别

标准的§6.7.8.2定义了一组基础系统架构，用于设计实现其SRCF的SRECS。“子系统”的定义是重点（参见§3.2.5），它被定义为其危险故障可导致安全功能丧失的系统架构部件。

这里以微控制器为重点，对IEC 62061推荐架构进行了简要总结，以便为终端用户开发可用作子系统（用于SRCF实现）的逻辑运算器单元提供支持。

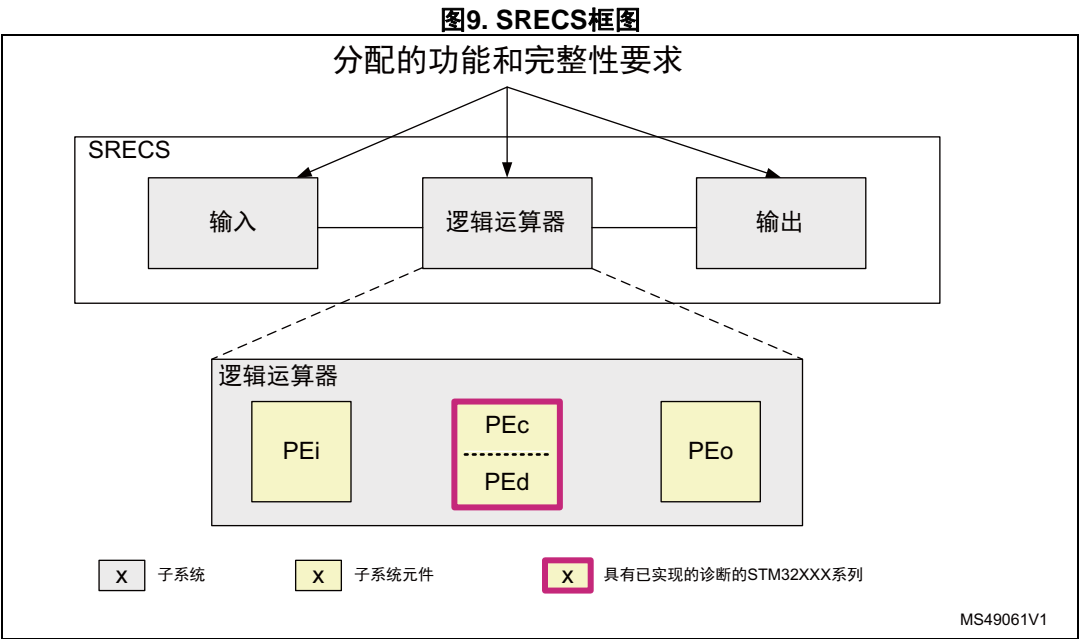
正确理解架构所需的假设如下：

1. SRCF完全处于终端用户的范围之内。
2. 采用本安全手册中所述安全机制的STM32L0系列器件作为单个合规项，适合最高SILCL 2级的应用。
3. 如果基础架构需要，为使 $HFT \neq 0$ ，必须使用两个相同的STM32L0系列器件并采用本手册中描述的安全机制。
4. 对于微控制器，在标准中作为使用寿命或验证测试之间的最小值提及的参数T1旨在作为生命周期（任务时间），假定等于10年（根据[第 3.3.1 节：假定安全要求](#)）。

表120. IEC 62061架构分类

仅限	参考 §	总结	基础逻辑架构
A	6.7.8.2.2	相当于1oo1, HFT=0, 无诊断功能。 总 PFH_{DSSA} 是MCU发生危险故障的概率。	单通道架构, 1oo1中1个MCU, $n = 1$ $PFH_{DSSA} = \lambda_{De1} \left[\frac{1}{\text{Hours}} \right]$ – SILCL = 1, 当SFF < 90%时 – SILCL = 2, 当 $90 \leq \text{SFF} < 99\%$ 时 – SILCL = 3, 当 $\text{SFF} \geq 99\%$ 时
B	6.7.8.2.3	相当于1oo2, HFT=1, 单一故障不会导致SRCF丧失。 无诊断功能。	双通道架构, 具有两个相同MCU – SILCL = 1, 当SFF < 60%时 – SILCL = 2, 当 $60\% \leq \text{SFF} < 90\%$ 时 – SILCL = 3, 当 $\text{SFF} \geq 90\%$ 时 在这种情况下: $\lambda_{De1} = \lambda_{De2} = \lambda_{De}$ $\lambda_{DSSB} = (1 - \beta)^2 \times \lambda_{De}^2 \times T_1 + \beta \times \lambda_{De}$ 关于 β 因子, 请参见第 4.2 节
C	6.7.8.2.4	相当于具有诊断功能的1oo1d, 此功能在SRCF上发生危险故障时触发反应功能。 注: 诊断功能提供具有外部子系统诊断 (例如, 执行器) 能力的逻辑运算器。	单通道架构, 1oo1中1个MCU, $n = 1$ 诊断功能由终端用户负责 – SILCL = 1, 当SFF < 90%时 – SILCL = 2, 当 $90 < \text{SFF} < 99\%$ 时 – SILCL = 3, 当 $\text{SFF} \geq 99\%$ 时 $\lambda_{DSSC} = \lambda_{De1} (1 - DC_1)$ 从FMEDA得出DC (诊断覆盖率) $PFH_{DSSC} = \lambda_{DSSC} \left[\frac{1}{\text{Hours}} \right]$
D	6.7.8.2.5	任何单一故障都不会导致SRCF丧失; 它相当于HFT = 1并具有诊断功能的1oo2d。 注: 诊断功能提供具有外部子系统诊断 (例如, 执行器) 能力的逻辑运算器	双通道架构, 具有两个相同MCU 诊断功能由终端用户负责 – SILCL = 1, 当SFF < 60%时 – SILCL = 2, 当 $60\% \leq \text{SFF} < 90\%$ 时 – SILCL = 3, 当 $\text{SFF} \geq 90\%$ 时 关于 β 因子, 参见第 4.2 节 从FMEDA得出DC (诊断覆盖率) 在这种情况下: – $\lambda_{De1} = \lambda_{De2} = \lambda_{De}$ – T2必须由终端用户在逻辑运算器层面进行定义

基于IEC 62061 §6, 图 9显示了如何通过实现该标准的图B.1中描绘的通用控制架构来开发SRECS, 这里显示的微控制器是采用了第 3.7节: 使用条件中定义的安全机制的STM32L0系列器件。



A.2.2 安全指标计算

T期间的故障率 (λ) 是验证测试间隔与子系统寿命之间的较小者。

如ISO 13849所示, §6.7.8.2.1注释2中的逼近值仍被视为有效, 因此 $\lambda = 1 / \text{MTTF}$, 其中, 假设 $1 \gg \lambda \times T$ 。

因此, 当 $\text{PFH}_D = \lambda_D \times 1\text{h}$ 时, $\text{PFHD} = 1 / \text{MTTF}$ 。

按照IEC61508对STM32L0系列执行的安全分析在危险故障识别的定义 (可以在IEC 62061的范围内重新映射) 方面越来越准确。因此, FMEDA中报告的 λ 和PFH值 (参见第 4节: 安全结果) 仍然有效, 可以用在上一段的公式中。

无需重新计算微控制器的SFF。终端用户使用从本安全手册中得出的相同值。

如前文第 4.2节: 从属故障分析所述, 在评估 $\text{HFT} = 1$ 的基础架构的CCF时, 终端用户使用通过IEC 61508方法获得的相同结果 (如果有) (参见IEC 61508:2010-6附录D)。或者, 终端用户可以应用经过简化的标准方法 (参见附录F), 以便计算要在PFHD公式中使用的 β 因子值。

A.2.3 工作成果

表 121列出了IEC 62061标准要求的工作成果以及它们与IEC 61508合规项工作成果的对应关系:

表121. IEC 62061工作成果列表

IEC 62061 1.1表8		STM32L0系列 IEC 61508文档
待提供的信息	IEC 62061-1.1条款	
功能安全计划	4.2.1	终端用户责任
SRCF要求说明	5.2	
SRCF的功能安全要求说明	5.2.3	
SRCF的安全完整性要求说明	5.2.4	
SRECS设计	6.2.5	STM32L0系列 安全手册
结构化设计过程	6.6.1.2	终端用户责任
SRECS设计文档	6.6.1.8	
功能块的结构	6.6.2.1.1	
SRECS架构	6.6.2.1.5	STM32L0系列 安全手册
子系统安全要求说明	6.6.2.1.7	终端用户责任
子系统实现	6.7.2.2	
子系统架构（元件及其相互关系）	6.7.4.3.1.2	STM32L0系列 安全手册
估计容错或SFF时声明的故障排除	6.7.6.1c / 6.7.7.3	终端用户责任
软件安全要求规格	6.10.1	
基于软件的参数化	6.11.2.4	
软件配置管理项	6.11.3.2.2	
软件开发工具的适用性	6.11.3.4.1	
应用程序文档	6.11.3.4.5	
应用软件模块测试结果	6.11.3.7.4	
应用软件集成测试结果	6.11.3.8.2	
SRECS集成测试文档	6.12.1.3	
SRECS安装文档	6.13.2.2	
安装、使用和维护文档	7.2	
SRECS确认测试文档	8.2.4	
SRECS配置管理文档	9.3.1	

A.3 IEC 61800-5-2:2007

该标准的范围是调速电气传动系统的功能安全。IEC 61800第5.2部分在IEC 61508第一版的框架内，定义了对设计、开发、集成和验证传动速度应用PDS（SR）的安全相关部件的要

求。更准确地说，IEC 61800的这一部分将其应用限制为在HD或CM中工作的PSD（RS）（参见§3.10注释1），它们实现目标完整性最高为SIL 3的安全功能。

从架构的角度来看，此限制反映在两个表格中，它们是§6.2.2.3的表 3和表 4，分别适用于两种不同类型的器件。CPU或整个微控制器（复杂电子部件）被划分为B类。此外，HFT的概念实际上也来自于IEC 61508。

A.3.1 架构类别

从架构的角度来看，IEC 61800应用反映在两个表格中，它们是§6.2.2.3的表 3和表 4，分别适用于两种不同类型的器件。由于CPU或整个微控制器被视为复杂电子部件，所以被划分为B类。此外，HFT的概念实际上也来自于IEC 61508。因此，架构重新映射到IEC61508十分直接。

A.3.2 安全指标计算

PDS（SR）执行的安全功能的PFH由IEC61508-2评估。在IEC 61800-5-2中，与IEC 61508规范的密切相关还反映在采用相同的重要指标PFH（参见§6.2.1）和SFF（参见§6.2.3）上。因此，可以在IEC61800的范围内重新映射本安全手册（和相关的FMEA或FMEDA）的结果。

A.3.3 工作成果

[表 122](#)列出了IEC 61800-5-2标准要求的工作成果以及它们与IEC 61508合规项工作成果的对应关系。

表122. IEC 61800工作成果列表

IEC 618000 5.2		STM32L0系列 IEC 61508文档
待提供的信息	IEC 61800-5.2部分-条款	
PDS（SR）的安全要求说明（SRS）包括安全功能要求 和安全完整性要求	5.4	终端用户责任
PDS（SR）安全要求说明的验证	8.2	
架构层面的硬件设计	6	
架构层面的软件设计	IEC 61508-3	
功能框图层面的随机硬件故障导致安全功能故障的概率 估计	IEC 61508-2	STM32L0系列 安全手册和FMEDA
系统设计审核	8.2	终端用户责任
安全相关PDS（SR）的详细确认规划	8.3	
硬件设计	6	
软件设计		
可靠性预测	6	STM32L0系列 安全手册和FMEDA

表122. IEC 61800工作成果列表（续）

IEC 618000 5.2		STM32L0系列 IEC 61508文档
待提供的信息	IEC 61800-5.2部分-条款	
系统设计审核	8.2	终端用户责任
模块层面的功能测试		
安全相关PDS（SR）的集成和测试。	6.5	
硬件或软件集成测试结果和文档的审核	8.2	
开发描述PDS（SR）安装、调试、操作和维护的用户文档。	7	
完整软件和相应文档	8.3	
确认测试结果文档		
符合确认计划的确认测试和程序		
确认测试结果文档		
子系统测试计划	6.2.4.1.4	
集成测试计划		
确认测试计划		
配置测试计划		
每项测试的详细结果	9.2.g)	
预期值与实际值之间的任何不一致	9.2.h)	
测试结论：通过或失败原因	9.2.i)	

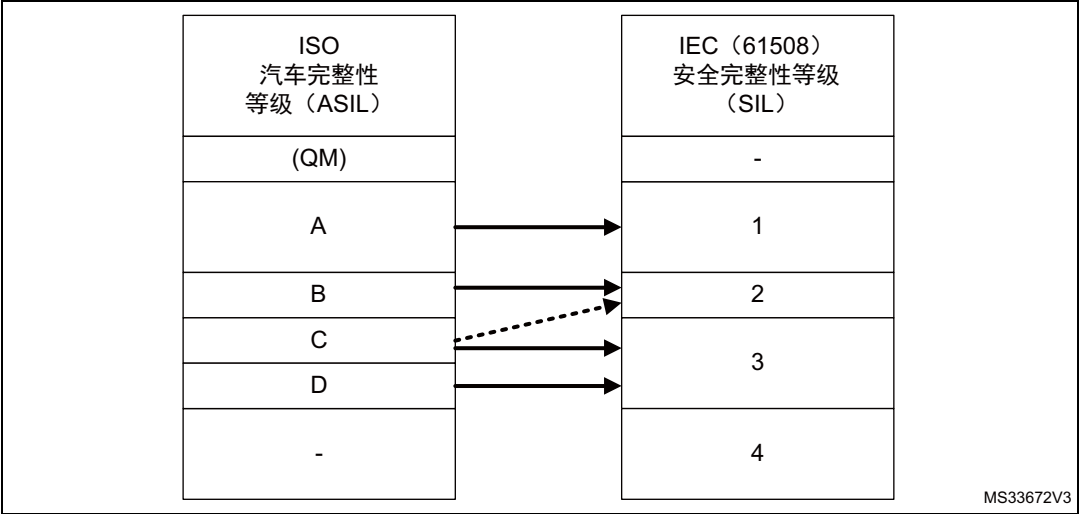
A.4 ISO 26262:2010

该国际标准是汽车领域功能安全的参考标准。它源于IEC 61508标准，包括相关修改。

ISO 26262重新定义了汽车SIL（ASIL）的安全完整性等级，最低等级为A，最高等级为D。
[图 10](#)描述了已按照TÜV SÜD进行实证的SIL和ASIL值之间的相关模型。



图10. SIL和ASIL之间的相关模型



A.4.1 架构类别

不适用 - 因为ISO 26262没有定义任何类别。

A.4.2 安全指标计算

ISO 26262标准从稍微有别于IEC61508标准的角度定义了硬件指标：

- 单点故障指标（SPFm）：使用IEC61508中SFF的相同公式定义，可能因安全故障的不同定义而有所不同（参见下文）
- 诊断覆盖率（DC）的定义方式与IEC61508的相同；
- 潜在故障指标（LFm）：专用ISO26262安全指标，用于评估在发生影响诊断部件的故障时设计的稳健性。IEC61508标准中无对等指标。

值得注意的是，一些故障在IEC 61508标准中被分类为无部件/无影响，而在ISO26262标准中被分类为“安全故障”。因此，IEC61508标准中的SFF计算是“保守的”，可以使用来自STM32L0系列FMEDA的SPF值。

对于此类商用现成品（COTS）微控制器（例如，STM32L0系列），ISO方案的指标是ASILB（永久和瞬时故障的SPF指标为90%，潜在故障为60%）。由于这些指标与1oo1 SIL2方案的相同，可以假定同一组使用条件或安全机制适用。STM32L0系列微控制器的FMEDA详细描述了指标计算；请注意，得到的PMHF值符合ASIL B MCU的期望值。

我们可以得出结论，ASIL B的目标可通过对最终应用施加一些限制来实现。请注意，安全诊断措施的基础是在每个FTTI定期执行软件至少一次。

对于STM32L0系列器件，通过采用保证微控制器适用于SIL2应用的相同安全机制组合，可以满足ASIL B潜在故障指标（60%）。

注：由于IEC61508和ISO26262在微控制器模块或功能的局部目标解读上存在差异，SIL2方案中通过STM32L0系列实现的安全性能与基于ISO26262-5第9.4.3节的ISO26262应用无法兼容（即“割集”法）。如果您的ISO26262安全分析使用此类方法，请仔细检查STM32L0系列FMEDA功能层面的故障率。

A.4.3 工作成果

表 123列出了ISO 26262标准要求的工作成果以及它们与IEC 61508合规项工作成果的对应关系：

表123. IEC 26262工作成果列表

IEC 26262		STM32L0系列 IEC 61508文档
待提供的信息	IEC 26262部分-条款	
技术安全要求规格	4-6.5.1	STM32L0系列 安全手册
技术安全理念	4-7.5.1	
从要求得出的安全分析报告	4-7.5.6	
硬件安全要求验证报告	5-6.5.3	
硬件安全分析报告	5-7.5.2	
处理随机硬件故障的项目架构的有效性分析	5-8.5.1	
处理随机硬件故障的项目架构的有效性评估审核报告	5-8.5.2	
随机硬件故障所致安全目标违规的分析	5-9.5.1	STM32L0系列FMEDA
随机硬件故障所致安全目标违规的评估审核报告	5-9.5.3	
软件安全要求规格	6-6.5.1	终端用户责任
软件架构设计规格	6-7.5.1	
软件验证报告（精简）	6-11.5.3	
安全分析结果	9-8.5.1	STM32L0系列 安全手册，FMEA和FMEDA

注：应改写STM32L0系列FMEA以便将IEC61508参考故障模式映射到ISO26262参考故障模式中。



版本历史

表124. 文档版本历史

日期	版本	变更
2016年3月18日	1	初始版本。
2017年12月13日	2	将文件分类更改为限于ST公司。 更新了 第 1.1 节：目的和范围 和 第 3.6.3 节：内部 SRAM 。 更新了 图 10：SIL和ASIL之间的相关模型 。 对整个文档进行了少量文字修订。
2018年6月20	3	文档分类由限于ST公司变为公开发布。 更新了 引言 、 第 1.1 节：目的和范围 、 第 1.2 节：术语和缩略语 、 第 1.3 节：参考标准 、 第 3.1 节：引言 、 第 3.3.1 节：假定安全要求 、 第 3.6.7 节：通用同步收发器 (USART1/2/4/5) 、 低功耗通用异步收发器 (LPUART1) 、 第 3.6.16 节：高级、通用和低功耗定时器 (TIM2/3/21/22, LPTIM1/2) 、 第 3.6.17 节：通用输入/输出 (GPIO) - 端口A/B/C/D/E/F/G/H 、 第 3.7 节：使用条件 、 第 4.1 节：随机硬件故障安全结果 、 第 4.1.1 节：安全分析结果自定义 和 第 5 节：证据列表 。 更新了 第 3.6 节：硬件和软件诊断说明 子章节中的格式。 更新了 图 1：STMicroelectronics产品开发过程 、 图 2：合规项的定义 、 图 6：ISO 13849类的框图 B类和1类的框图 1 、 图 6：ISO 13849类的框图 B类和1类的框图 1 、 图 7：ISO 13849类的框图 2 、 图 9：SRECS框图 和 图 10：SIL和ASIL之间的相关模型 。 删除了原 章节2.2：Ygitech fRMethodology过程 、 图5：STM32L0模块的安全特性框图 。 第3.6.4 节：系统EEPROM 、 附录A：fRMethodology概述 和 附录D：fRSTL_STM32L0_SIL2/3产品及其在本手册框架中的使用 。 增加了 第 3.2.3 节：参考安全架构 - 1oo1 ， 第 3.2.3 节：参考安全架构 - 1oo1 ， 第 4.1.3 节：关于多故障情景的说明 以及 表 3：SS1和SS2安全状态详细信息 。 更新了 表 1：术语和缩略语 、 表 2：本文档与IEC 61508-2附录D要求之间的对应关系 、 表 22：RAM_SM_0 、 表 40：UART_SM_3 、 表 62：ADC_SM_4 、 表 108：AES_SM_2 、 表 114：安全机制列表 ， 表 115：整体可达到的安全完整性等级 和 表 120：IEC 62061架构分类 。 对整个文档进行了少量文字修订。

表125. 中文文档版本历史

日期	版本	变更
2019年5月28日	1	中文初始版本。

重要通知 - 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用，ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。若需 ST 商标的更多信息，请参考 www.st.com/trademarks。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。

© 2019 STMicroelectronics - 保留所有权利