AN1015
应用笔记

提高微控制器 EMC 性能的软件技术

前言

在基于微控制器的电子系统中，强壮的软件设计是提高 EMC 性能的主要因素。

必须尽量在项目设计阶段的早期考虑 EMC 干扰导致的问题。 EMC 导向的软件提高了应用的安全性和可靠性。强化 EMC 性能的软件的实施成本低，可提高最终的抗扰性能，并节约硬件和开发成本。用户应考虑到模拟或数字数据受到的 EMC 干扰，就像任何其它应用参数一样。

EMC 干扰导致的问题示例：

- 微控制器无响应
- 程序计数器失控
- 执行意外指令
- 地址指向错误
- 子程序执行错误
- 寄生复位和 / 或寄生中断
- IP 配置损坏
- I/O 失灵

软件故障的后果示例：

- 产品意外响应
- 上下文丢失
- 进程中的意外分支
- 中断丢失
- 数据完整性缺失
- 输入值误读

本应用笔记将介绍以下两类软件技术：

- 预防技术：它可以在已有的设计中实现，其目的是提高产品的稳健性。
- 自动恢复技术：当检测到失控情况时，将使用恢复子程序来决定执行故障安全程序，也可以选择发出警告，然后自动恢复为正常运行状态。（此操作对于应用的用户可能是完全透明的）。

目录

表格索引

图片索引

1 相关文档

- AN3181“ 在 STM8 的应用中获得 IEC 60335 类型 B 认证指南 ”
- AN3307“ 在 STM32 的应用中获得 IEC 60335 类型 B 认证指南 ”
- AN4435“ 在 STM32 的应用中获得 UL/CSA/IEC 60335 类型 B 认证指南 ”

2 预防技术

用户可以在已有的设计中实施预防技术，用以提高产品的稳健性和对外部或内部 EMC 干扰的抗扰性。

2.1 使用看门狗和时间控制技术

为了确保 MCU 能够从软件失控故障中恢复，看门狗是最有效的可用工具。它的原理非常简单：它是一个在计数结束时产生 MCU 复位的定时器。一旦看门狗启动，防止看门狗复位控制器的唯一方式就是在程序中周期性更新计数器。

但是为了使看门狗发挥它的最大潜力，用户必须在软件中的恰当位置插入使能和刷新指令。

图 1 显示了两种典型的看门狗使用不当的示例。

为了以正确的方式实现 （参见*图 2*），需要执行下列规则：

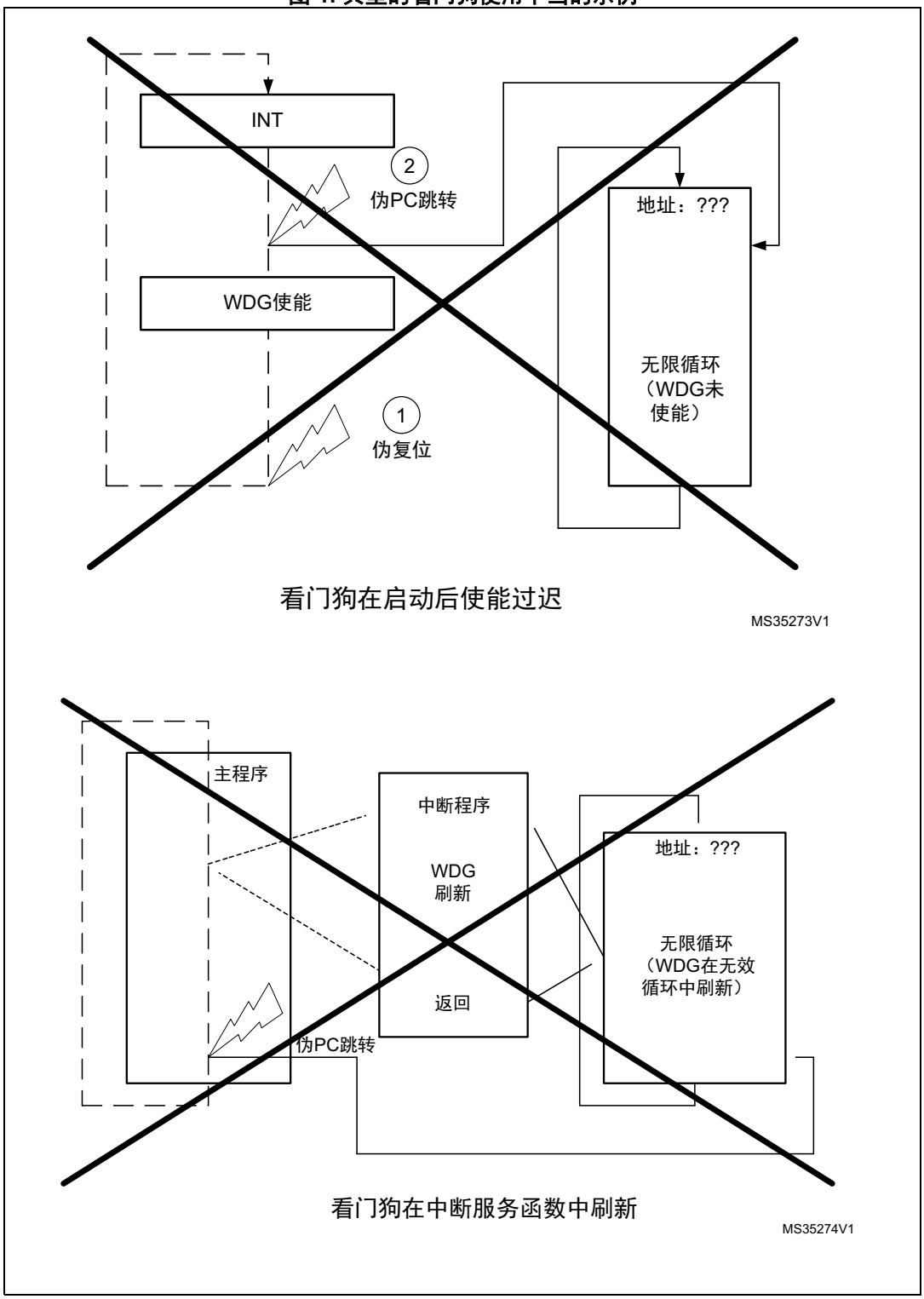
- 复位之后尽快启用看门狗，或者开启硬狗选项 （如果有）。
- 切勿在中断服务函数中或在代码中没有超时保护的任何本地环路中刷新看门狗。

根据不同程序的持续时间，包括中断程序，对两个刷新指令之间的间隔时间进行优化是非常重要的。

看门狗还有一种小用途，即可以使用它来复位 MCU，但这也意味着程序上下文的丢失，以及应用数据完整性的丢失。

复位之后，除了启用看门狗外，在一些 MCU 上，你可以使用复位标志来区分上电或低电压复位或看门狗复位 （参见*第 3.3 节：使用复位标签辨识复位源* 获取更多信息）。

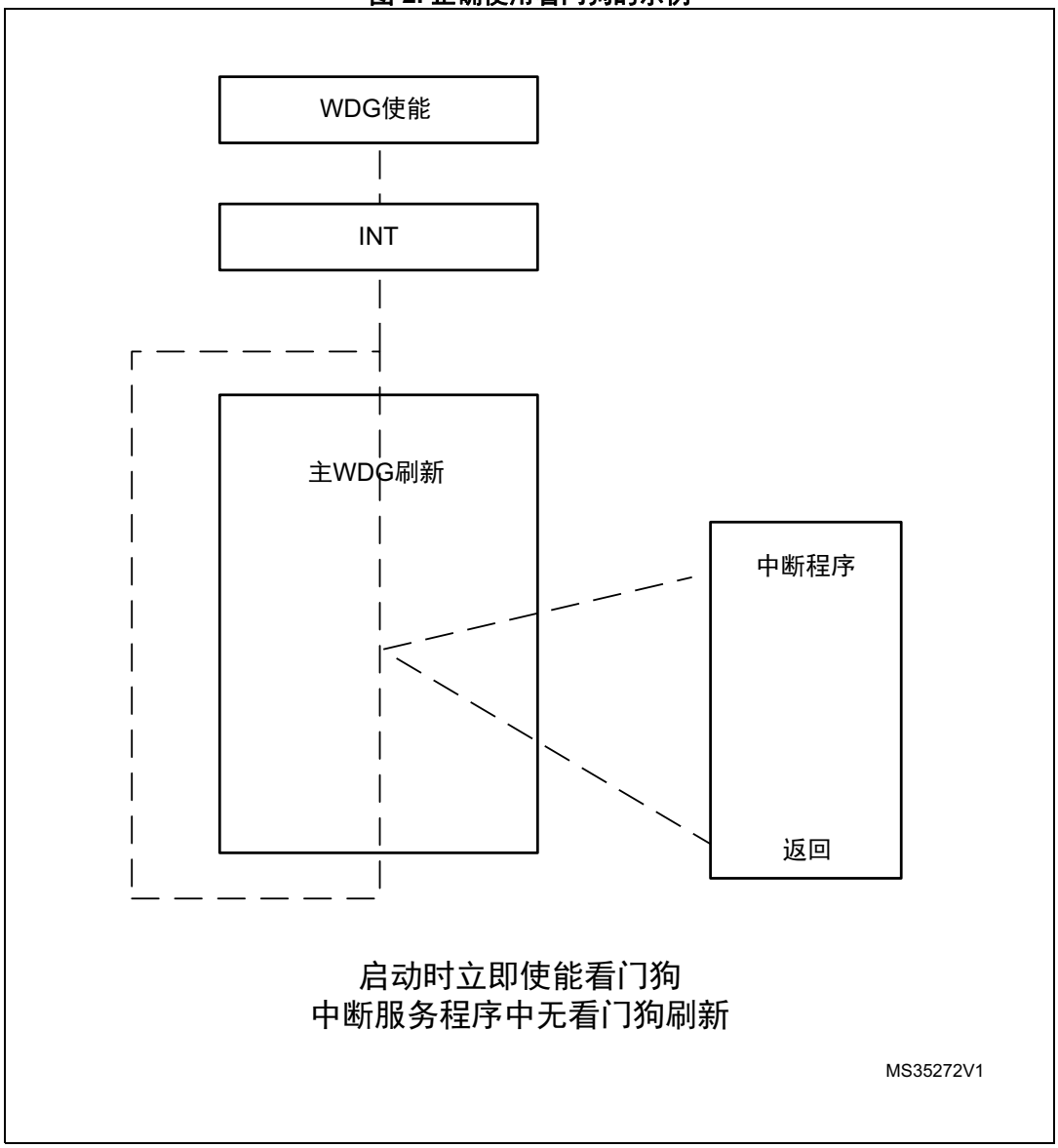
图 1. 典型的看门狗使用不当的示例



看门狗在启动后使能过迟

看门狗在中断服务函数中刷新

图 2. 正确使用看门狗的示例



2.2 保护闲置程序存储区域

在大多数应用中，程序存储空间并没有完全被用户代码占满。为了更强的安全性，如果你不想产生复位，可以使用代码填满闲置存储位置，这样会强制看门狗复位或者跳向已知的程序位置。

即使程序计数器损坏，并且跳向闲置存储位置，这都将确保 MCU 恢复和返回到正常操作。

在这个闲置区域，还可以跳转到恢复故障安全子程序，通过该子程序可恢复正常运行。

对于 STM8 用户，为了从跳向意外存储区域中恢复，用 STM8 "TRAP" 指令 （只有一个指令字节：83）生成软件中断也很方便。

另一个有效的恢复正常操作的方式是用非法指令操作代码的值填满存储器，该代码可以在取址和执行时复位 STM8 微控制器。

带有 ARM® Cortex®-M 内核的 STM32 微控制器使用故障异常，当系统遭受 EMC 干扰时，它可以捕获可能发生的非法存储访问和非法编程行为。未定义的指令操作代码可以用来填充 STM32 微控制器的闲置存储，在程序计数器失控的情况下，提高故障异常使用率，使得故障安全程序从错误中恢复。

另外的选择是使用通过 SVC 指令调用的系统服务来执行故障安全程序。

2.3 输入过滤和比较

最佳做法是在验证状态和继续编程执行之前，由程序对输入引脚稳定性进行几次检查，以避免由输入电路的外部噪声造成的不想要的尖峰响应。

这是一个简单的对关键输入滤波的方法，不需要额外的成本！

2.4 闲置中断向量管理

为了避免意外中断事件导致的问题 （无论是什么来源），建议在相应的向量中加入一个有效的中断程序地址，以管理所有可能的中断来源。

在下面的示例中，未使用的中断向量指向故障管理程序标签，该标签中填充了简单的 “ 从中断返回 ” 指令。

2.5 从代码中去除非法和关键字节

2.5.1 关键字节

关键字节是一个在低功耗模式下开关 MCU 的指令，它由微控制器解码，并且强迫使微控制器停止执行任何指令。

当 PC 损坏时，微控制器经常变得不同步 （因为大多数指令都有几个字节），结果它可能会读和解码关键字节。

为了检查和减少关键字节的出现，你可以检查程序的 ".list" 文件。

通常关键字节由编译器生成，并作为标签地址字节。在这种情况下，如果你简单地插入一个或几个 NOP 指令，所有的标签地址将会移位，同时这将会改变关键字节的值。

2.5.2 非法字节

非法字节定义为任何不属于指令集的字节值。它们要么作为 NOP 指令执行，或者 （在一些 MCU 上）如果遇到一个非法字节，就会生成一个复位。在这种情况下，使用上面描述的技术 （用于关键字节）从你的代码中去除非法字节。

2.6 平均模数转换器结果

如果你在执行 A/D 转换操作，你可以重复多次转换，将结果存储在 RAM 中，然后取平均值（或者选择最多出现的值），以在任何可能的噪声错误的情况下获取准确的结果。

2.7 寄存器重编程和常规检查

很少会出现 EMC 干扰使寄存器内容发生改变的情况。通常关注到的寄存器包括时钟控制寄存器或 I/O 配置和数据寄存器，因为它们靠近芯片的输出焊盘。

在这种情况下，一个不错的安全方法是经常刷新这些寄存器。

表 1. 预防技术总结

软件质量 预防技术	优点	缺点	实现方法
看门狗 （硬件或软件）	控制不依赖 CPU 避免 MCU 锁定	如果使用 LP 模式，需要仔细处理	简单但是激活和刷新指令必须在代码中仔细放置以达到最大的效率
在闲置程序存储器中强制加入看门狗复位	比等待看门狗超时更直接和快速	之前内容损失	清除看门狗复位比特（参见器件参数）
用软件中断指令填充闲置程序存储器	比等待看门狗超时更直接和快速。	无	用 "TRAP" 或 SVC 操作代码填充闲置区域，同时管理相应中断程序的故障。
模数转换器平均	在噪声环境中保证 ADC 的性能。	处理时间	执行迭代循环用于 ADC 采集和平均。
去除非法或关键操作代码	避免由于意外 WFET 或 WFI 操作代码导致的 MCU 锁定	无，除了对使用这些操作代码的限制	在 ".list" 文件中寻找关键字符 （参见*第 2.5 节*）。
输入滤波	数据采集稳定性	处理时间	重复多次测量，同时执行 “0” 或者 “1” 之间的统计选择
闲置中断管理	避免由于意外中断导致的失控	无	非常简单 （参见*第 2.4 节*）
关键寄存器刷新	安全运行	使用 MCU 资源	在常执行环路中刷新关键寄存器

2.8 冗余数据存储和交换

由于极端条件下的电磁干扰，所有存储在内部或外部存储器的数据都可能损坏。

高级的预防技术包括将双倍互补值存储在非临近存储区域、存储和检查奇偶校验位或 ECC，它们可帮助识别和 / 或修正数据损坏。

更多关于提高软件稳健性的技术，请参考 AN4435 或安全手册，可以从 st.com 网站上获取。

3 自动恢复技术

本章给出了一些在 EMC 故障之后，快速恢复应用程序的技术。

不管干扰的来源是什么，意外复位、程序计数器跳转和寄生中断是 MCU 中最常见的 EMC 故障。

在任何情况下， RAM （或者 EEPROM 数据存储器）保持不变，可以用来作为非常有效的保存应用内容和参数的方式。

注意到当器件掉电时， RAM 不会保存内容。 EEPROM 数据存储器会在掉电时保持内容，但是需要更长的写时间。

3.1 在 RAM 中保存你的内容

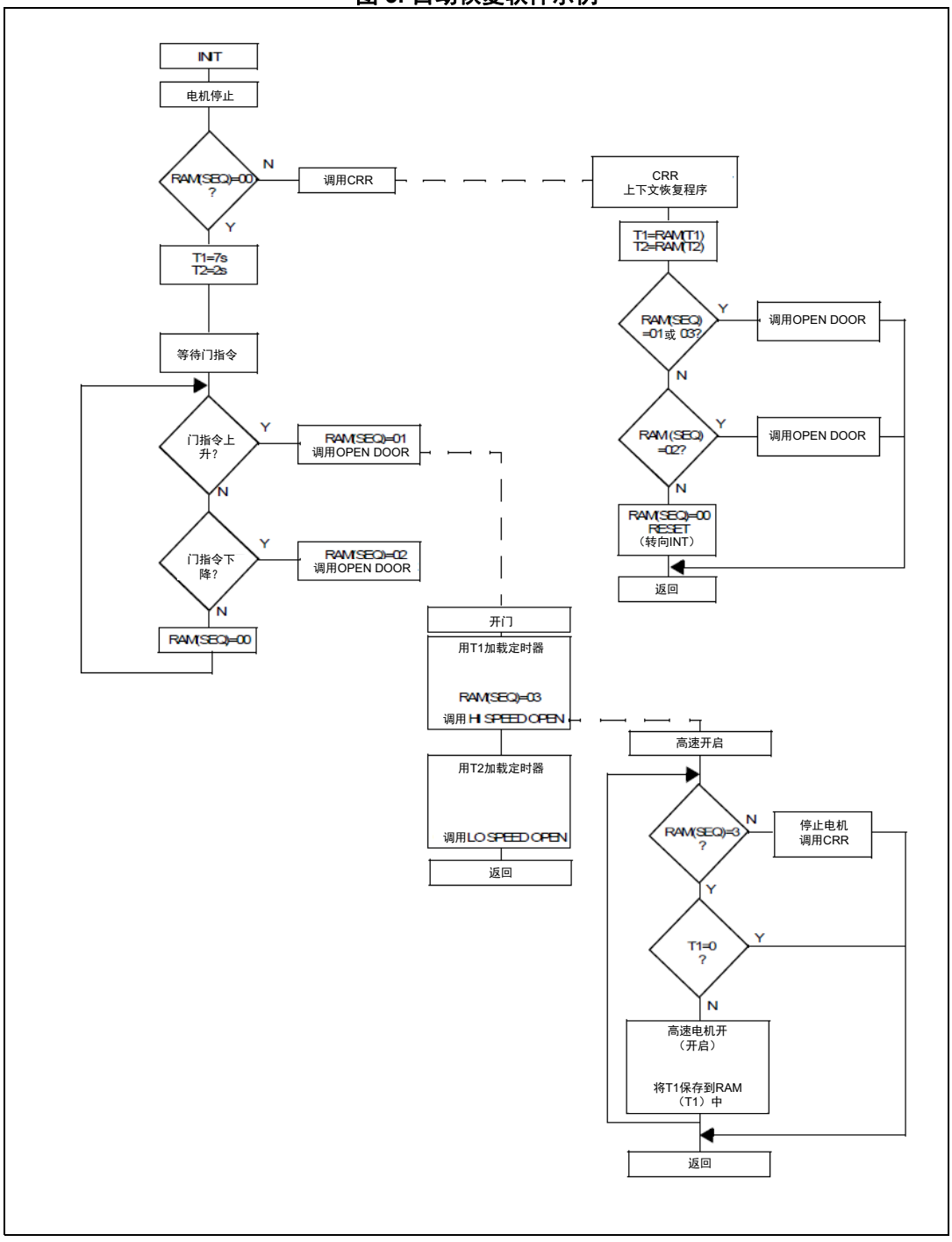
图 3 显示了一个软件自动恢复实现的示例：关键软件序列 （门 OPEN 或 CLOSE 指令、高速电机控制）存储在 RAM 字节中 ("RAM(SEQ)")。

如果 EMC 事件导致 MCU 复位，这使得我们一方面可以恢复内容，另一方面我们可以在执行关键子代码之前检查来源。在这种情况下，只有当 RAM(SEQ)=03) 时，高速电机才会激活。

当应用参数 （T1&T2 时序值）改变时，它们也存储在 RAM 中。

这意味着如果软件失控事件发生或者 MCU 复位 （通过低电压监测或看门狗），恢复程序 (CRR) 将会存储最后的门命令、重新载入时序参数和恢复程序执行，不需要任何外部干预。

图 3. 自动恢复软件示例

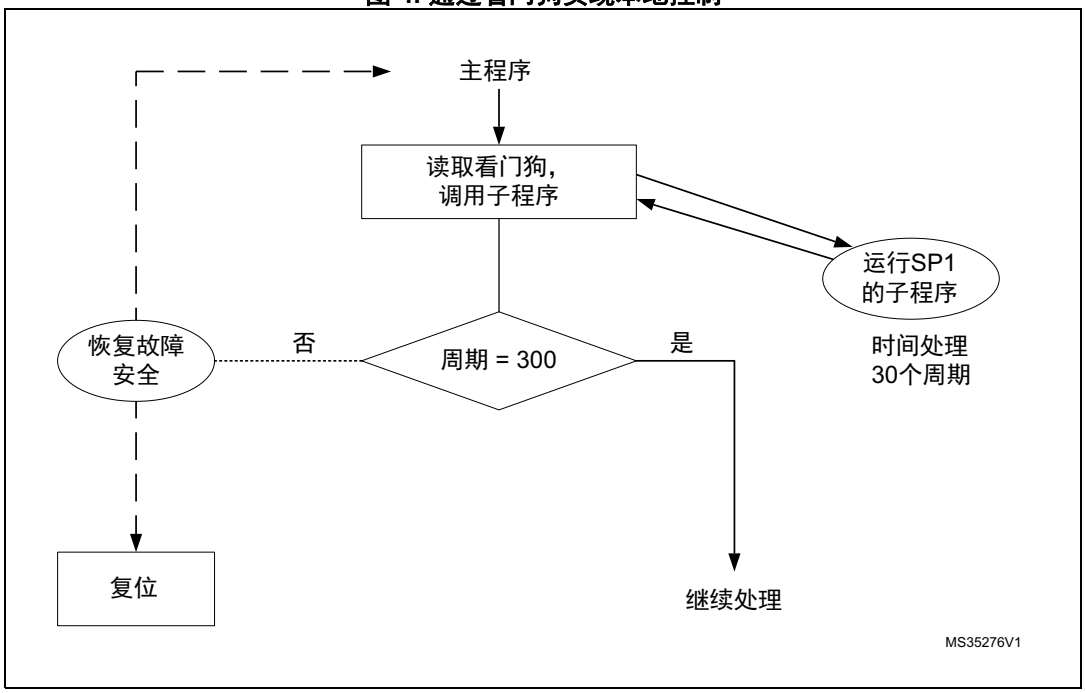


3.2 在本地控制中使用看门狗

编程人员常常将看门狗定时器视为一个定时炸弹，而且仅将其刷新为最大值，以获得尽可能宽的裕量，并与期望的程序执行时间没有任何直接关系。

这是一个不好的方式，更好的方法是使用看门狗定时器寄存器检查单独软件程序的执行时间，如果在看门狗结束计数前出现异常，则迅速做出反应，执行立即复位或转到软件恢复程序。

图 4. 通过看门狗实现本地控制



3.3 使用复位标签辨识复位源

内部复位源可能包括：LVD （低电压检测器）或看门狗复位、 POR （上电复位）、热复位（在 Reset 引脚的低电平状态后的寄生或外部复位）。

复位源标记在一个 “ 复位寄存器 ” 中，只要 MCU 处于上电状态，该信息就会一直保存。

图 5 显示了如何在程序开始时测试复位寄存器，然后分支到上下文恢复程序 （取决于检测到的复位源），而不是重新启动通常很复杂且耗时的 “ 上电复位 ” 初始化程序。

检测和管理寄生复位是很重要的，这是因为它们是微控制器 EMC 故障最常见的原因。

图 5. 识别复位源

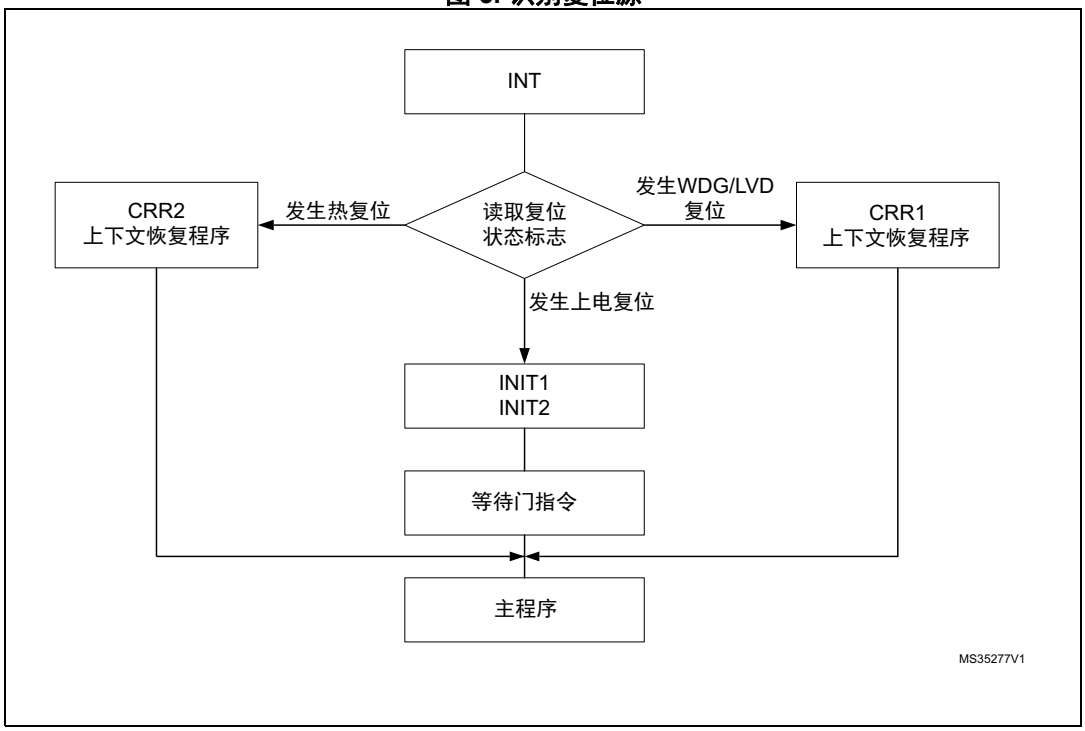


表 2. 自动恢复技术总结

软件质量 自动恢复技术	优点	缺点	实现方法
通过看门狗实现本地控制	关键连续块的过程控制	需要计算准确的时间窗口	使用 WDG 定时器寄存器检查序列执行时间
识别复位源	从意外复位故障中快速恢复	无	使用 MCU“ 复位寄存器 ” 或 RAM 检测各种复位源
应用上下文保存在 RAM、 FLASH 或 EEPROM 中	保存应用参数，确保在 MCU 发生故障时恢复关键任务执行	使用 MCU 资源	软件关键状态和参数存储在 RAM、 Flash 或 EEPROM 中。 使用 RAM、 Flash 或 EEPROM 中的数据恢复故障之前的最后内容。

3.4 数据存储在非易失性存储器中

数据存储在非易失性存储器 （数据 EEPROM）所需的时间明显长于数据存储到 RAM 的时间。在编程期间， EMC 干扰可能会危及到系统，导致系统复位终止程序运行，从而造成数据损坏。

为了防止这种情况出现，数据应该存储在冗余存储器中，通过使用有效标记保持它的一致性。

在每个系统启动之后，实际使用之前，都需要检查数据存储器中内容的有效性。

4 能获得什么结果？

经过设计、测试和优化的 ST 微控制器可以在 ESD 电压 （根据 EN1000-4-2 标准）直接加到任何引脚时仍然保持完全的功能 （电平值参见其数据手册）。尽管在大多数情况下，这样的性能都已足够好，但同时还可以使用软件技术提高性能，以确保系统能在 4kV 的 EMC 电平时正常响应。

正确设计的系统可以检测由 EMC 干扰导致的损坏，启动并成功完成自动恢复程序，使得系统复位或重新初始化，这样的系统总是要好于不检测任何问题，不启动自动恢复机制，已部分损坏但没有任何可见变化的系统。

5 修订历史

表 3. 文档修订历史

日期	版本	变更
2001 年 7 月 2 日	1	初始版本。
2014 年 6 月 24 日	2	修改了*前言*、*第 2 节*、*第 2.1 节*、*第 2.2 节*、*第 2.3 节*、*第 2.4 节*、*第 2.5 节*、*第 3.2 节*和*第 4 节* 增加了*第 1 节：相关文档*、*第 2.8 节：冗余数据存储和交换*和*第 3.4 节：数据存储在非易失性存储器中*。 更新了*图 3* 和*图 4*。 更新了*表 1：预防技术总结*和*表 2：自动恢复技术总结*。

表 4. 中文文档修订历史

日期	版本	变更
2015 年 9 月 23 日	1	中文初始版本。