嵌入式 C 语言软件的代码优化与设备运行稳定性提升技术研究

俞光日
天津七一二通信广播股份有限公司，天津 300462
DOI:10.61369/ETQM.2026090040

摘　　要： 本文针对嵌入式 C 语言软件的代码优化与设备运行稳定性提升技术进行了系统性研究，从代码编译层级优化、代码逻辑与语法优化、内存资源优化和运行效率与功耗优化四个维度，详细阐述了嵌入式 C 语言的精细化优化技术。从软硬件协同稳定性优化、异常容错与故障自愈技术、中断与任务调度稳定性优化、数据运行与通信稳定性保障以及极端工况稳定性适配技术五个方面，分析了提升嵌入式设备运行稳定性的关键技术。通过理论分析与技术验证相结合的方式，提出了多种实用优化方案，为嵌入式系统开发提供了可靠的技术参考，有助于提升嵌入式系统的整体性能和长期运行稳定性。

关 键 词： 嵌入式 C 语言；代码优化；稳定性提升；内存管理

Research on Code Optimization and Device Stability Improvement Techniques for Embedded C Language Software

Yu Guangri
Tianjin 712 Communication and Broadcasting Co., Ltd., Tianjin 300462

Abstract: This article systematically studies the code optimization and device stability improvement techniques for embedded C language software. It elaborates on the refined optimization techniques of embedded C language from four dimensions: code compilation level optimization, code logic and syntax optimization, memory resource optimization, and running efficiency and power consumption optimization. The key technologies for improving the operational stability of embedded devices were analyzed from five aspects: software and hardware collaborative stability optimization, abnormal fault tolerance and self-healing technology, interrupt and task scheduling stability optimization, data operation and communication stability guarantee, and extreme working condition stability adaptation technology. By combining theoretical analysis with technical verification, multiple practical optimization solutions have been proposed, providing reliable technical references for embedded system development and helping to improve the overall performance and long-term stability of embedded systems.

Keywords: embedded C language; code optimization; stability improvement; memory management

引言

嵌入式系统作为计算机技术的一种应用形式，已广泛应用于工业自动化、消费电子、医疗设备、智能交通等领域。随着物联网和人工智能技术的快速发展，嵌入式系统的规模和复杂性不断增加，对其性能、功耗和可靠性的要求也越来越高。然而，嵌入式系统通常面临资源受限（如有限的 ROM/RAM）、实时性要求严格、硬件耦合紧密等挑战，这使得代码优化和稳定性保障成为嵌入式开发的核心议题。本文基于多年嵌入式项目实战经验，系统梳理了嵌入式 C 语言代码优化和稳定性提升的关键技术。所有方法均经过 ARM Cortex-M 系列、AVR、MSP430等主流 MCU 平台验证，聚焦工程可实现性，旨在为嵌入式开发者提供一套系统、实用、高效的优化方案。

一、嵌入式 C 语言代码精细化优化技术研究

（一）代码编译层级优化

在对嵌入式系统进行研究与设计的过程中，需要通过变量的使用对其进行有效的分析，以此实现对系统的完整架构，进而提升 C 代码的优化质量[1]。编译层级优化是嵌入式 C 语言代码优化的核心基础，可在不改动业务逻辑的前提下，精简机器指令、缩减代码体积、提升运行效率，适配嵌入式硬件资源受限的特点[2]。

 Copyright © This Work is Licensed under A Commons Attribution-Non Commercial 4.0 International License.

Keil、GCC、IAR 等主流嵌入式编译器，均配备多级适配不同场景的优化策略。基础优化安全性极高，主要清理冗余指令、无效代码与空循环，剔除未调用函数、未使用变量和无效逻辑分支，仅缩减存储占用，不改变程序运行逻辑，适用于所有嵌入式项目[3]。中级优化主打提升执行效率，通过常量折叠、指令重排、循环展开、小型函数内联等方式，消除函数堆栈调用开销、减少循环判断指令，简化执行流程。高级优化面向全局代码，通过冗余跳转删除、全局变量优化、适配指令流水线等实现效率最大化，但存在逻辑偏移的风险，需结合硬件调试配置。此外，需匹配 MCU 指令集、主频、缓存等硬件参数调整编译选项，精准适配硬件，保障代码高效、稳定运行。

（二）代码逻辑与语法优化

代码逻辑与语法不规范是嵌入式程序异常的主要人为诱因，开展精细化逻辑语法优化，能够从源头规避隐性故障，增强代码健壮性[4]。语法规范上，应统一编码标准，消除隐式类型转换、未初始化变量、指针越界、数组下标超限等基础问题；严控全局变量使用，过多全局变量会加剧代码耦合，易出现变量被篡改、逻辑紊乱等情况，优先选用局部、静态变量，保障变量访问安全。逻辑优化方面，简化复杂逻辑，削减多层 if—else 嵌套、冗余 switch 分支与无效循环判断，采用扁平化结构提升可读性与运行效率。循环中将固定运算移出循环体，不在循环内执行耗时运算、函数及硬件操作，设置超时退出机制，防范死循环、溢出与程序阻塞。函数遵循单一职责原则，控制代码规模，做好入参合法性校验。及时清除冗余、废弃及调试残留代码，降低版本迭代过程中的故障隐患。

（三）内存资源优化技术

嵌入式设备内存资源有限，内存操作不当极易造成设备死机、程序跑飞与数据异常，内存资源优化是提升系统稳定性的关键[5]。优化内存分配方式，尽量摒弃随意动态内存分配的开发习惯，优先采用静态内存分配，在编译阶段完成内存固定分配，规避运行时频繁 malloc/free 带来的内存碎片、分配失败及内存泄漏问题。确需使用动态内存的复杂场景，要统一内存分配接口，增设分配与释放校验机制，保证内存及时释放，防止泄漏。针对大量布尔型状态标志位场景，可利用 struct 位域特性对多个状态标志位进行捆绑存储，将多个标志位压缩至同一个字节内，避免每一个状态标志单独占用完整字节，有效降低 RAM 内存开销。对于只读的全局或静态常量，使用 const 关键字进行修饰，将常量数据存放至 Flash 存储空间，减少对宝贵 RAM 资源的占用。规范指针操作，防范野指针、空指针访问，指针使用前做好初始化和非空校验，指针偏移控制在合法内存范围，禁止指针跨类型强制转换[6]。合理设置数组长度，减少内存浪费，严格校验数组下标，防止数组越界破坏有效数据。做好堆栈管理，依据任务需求合理配置栈大小，不在函数内定义超大局部变量，防止栈溢出，监控堆栈使用率并优化代码降低占用。统一内存数据存储格式，规避数据对齐异常导致的读取错误，提升内存读写可靠性。

（四）运行效率与功耗优化

嵌入式设备多采用电池或工业低压供电，运行效率与功耗直接决定设备续航及长期运行稳定性，效率低、功耗高易造成发热、电压波动，诱发各类故障[7]。运行效率优化重在削减无效硬件占用与指令耗时，精简硬件操作流程，规避频繁硬件初始化、端口切换、重复采集等冗余操作；对于代码体量较小、调用频繁的函数，可使用 static inline 关键字实现函数内联，消除函数调用带来的堆栈压栈、跳转返回开销，缩短指令执行周期，进一步提升代码执行效率；精简高频核心代码，缩短指令周期，提升响应速度。多任务系统需优化任务时序，合理分配运行周期，低优先级任务采用轮询延时执行，防止抢占资源，保障高优先级实时任务运行。功耗优化遵循按需运行、闲置降耗，闲置时关闭闲置外设、时钟与通信模块，进入休眠；依据业务动态调节芯片主频，优化数据传输，减少无效收发校验；非实时业务拉长扫描周期，降低 CPU 无效运算，实现效率与功耗动态平衡，缓解设备发热老化，提升整体稳定性[8]。

二、嵌入式设备运行稳定性提升关键技术

（一）软硬件协同稳定性优化

嵌入式设备运行稳定性不只取决于软件代码，软硬件协同适配失衡是隐性故障的重要诱因，软硬件协同优化是提升设备稳定性的关键[9]。硬件侧需参照芯片与外设手册规范驱动开发，严格遵循硬件时序编写代码，保证软件指令与硬件时序匹配，防止时序错位造成响应异常、数据失真，串口、SPI、I2C 等外设要对齐波特率、时钟、延时等参数，规避通信故障。软件侧要结合硬件特性适配，依据硬件容错能力设置参数阈值，防止超量程指令损伤硬件；针对 ADC、时钟偏移等硬件固有误差，依靠滤波、校准算法做补偿修正。搭建软硬件状态联动，软件实时监控硬件，遇过载、过热、通信故障及时启动保护、复位外设，遏制故障蔓延。同时优化初始化逻辑，采用分层、延时初始化，增设初始化校验，失败则重试告警，从协同角度减少适配类故障。

（二）异常容错与故障自愈技术

嵌入式设备多部署于无人值守场景，难以人工实时干预，异常容错与故障自愈是保障设备长期稳定运行的关键[10]。异常容错面向数据、运算、硬件、通信等运行异常，通过代码层异常校验，预判拦截数据超限、参数非法、通信超时等风险，规避程序崩溃。依托软件看门狗监测程序状态，程序卡死、跑飞时自动软复位，防止设备瘫痪；针对瞬时干扰，设置重试机制，过滤假性故障，提升抗干扰能力。故障自愈实施分级处置：轻微故障通过刷新数据、重启任务修复；中度故障执行模块复位、重配资源；重度故障触发系统软重启恢复运行。配套故障记录，留存故障类型、时间及运行参数，支撑问题迭代优化，实现故障闭环治理。

（三）中断与任务调度稳定性优化

中断与任务调度是嵌入式系统核心逻辑，调度紊乱、中断处理不当易造成系统崩溃、任务超时与数据错乱。中断优化需恪守短中断、快响应、无阻塞，禁止在中断服务程序中执行动态内存分配，以及调用 printf 等调试打印类接口；中断服务程序访问的共享变量需要使用 volatile 关键字进行修饰，防止编译器做不当优

Copyright © This Work is Licensed under A Commons Attribution-Non Commercial 4.0 International License.

化造成数据异常。中断服务程序只做状态读取、标记赋值等轻量操作，将复杂运算与业务逻辑移出，防止中断耗时过久引发阻塞与嵌套溢出。按实时需求分级配置中断优先级，管控中断嵌套层数；临界代码段增加中断开关保护，规避中断打断造成的数据异常。RTOS 任务调度层面，合理配置任务优先级、周期与堆栈，防范优先级倒置、死锁及资源竞争。借助优先级继承处理优先级倒置，通过信号量、互斥锁、消息队列实现资源隔离。优化任务时序，规避高低频任务冲突，剔除无效空转任务，降低资源开销，保障多任务稳定高效调度。

（四）数据运行与通信稳定性保障

数据是嵌入式设备业务运行的核心载体，数据运算错误、存储异常、通信丢失会直接造成设备功能失效，需从运算、存储、通信三方面构建稳定性保障体系。

数据运行上，合理选型整型、浮点数据类型，规避溢出与精度丢失，增设运算超限判断；引入数据滤波算法，对传感器原始数据做去噪校准，保证运算准确。存储层面，结合 Flash、EEPROM 介质特性规范读写逻辑，减少高频读写损耗；核心数据实行双备份，通过 CRC 校验防范数据错乱篡改；配置断电检测与延时存储，防止断电引发数据保存不完整。通信方面，面向串口、CAN、以太网、LoRa 等场景，落实超时重传、帧过滤、断线重连机制，规范协议帧格式；启用超时检测，优化收发缓冲区，避免缓冲区溢出，保障数据收发完整可靠，减少通信卡顿、指令失效等故障。

（五）极端工况稳定性适配技术

嵌入式设备多用于工业户外、车载、井下等复杂场景，面临高低温、电压波动、电磁干扰、长期连续运行等极端工况，工况适配能力直接决定设备稳定性。

高低温上，依托代码优化适配硬件温度特性，低温增设硬件预热延时与数据等待机制，规避硬件响应延迟引发程序故障；高温实时监测温度，超标后执行降频、减负及告警保护，防止高温过载。电源方面，实时检测电压，电压低于阈值时完成数据保存并进入待机保护，防范断电造成数据丢失，采用软件滤波算法削弱电压波动带来的影响。电磁干扰场景下，优化代码逻辑减少高频操作，配合端口抗干扰配置、数据校验与多次采样，滤除干扰带来的假性异常。针对长期运行老化问题，优化内存调度，定期清理碎片、释放闲置资源，检测并自动修复轻微异常，缓解硬件老化损耗，全面提升设备工况适配与长效运行能力。

三、结束语

嵌入式 C 语言代码优化与设备运行稳定性提升是一个系统工程，需要从编译层级、逻辑语法、内存管理、运行效率等多个维度进行精细化优化，同时结合软硬件协同、异常容错、中断调度、数据通信和极端工况适配等关键技术，全面提升嵌入式系统的可靠性和稳定性。本文提出的技术方案均经过实际项目验证，能够在显著提升系统性能的同时，有效降低故障率，延长设备使用寿命。随着嵌入式系统向智能化、网络化方向发展，代码优化和稳定性保障技术将面临新的挑战。通过持续的技术创新和工程实践，嵌入式系统必将在更广泛的领域发挥重要作用，为万物互联的智能世界提供坚实的技术基础。

参考文献

[1] 薛亚，朱娅晶 . 嵌入式 C 语言系统开发代码优化研究 [J]. 电脑知识与技术 ,2020,16(29):227-228.

[2] 孟涛 . C 语言在嵌入式系统开发中的代码优化 [J]. 福建电脑 ,2008(3):181-182. DOI:10.3969/j.issn.1673-2782.2008.03.112.

[3] 陈波，石旭刚，史故臣 . 嵌入式 C 语言在系统开发中的代码优化 [J]. 计算机时代 ,2008(11):55-57. DOI:10.3969/j.issn.1006-8228.2008.11.023.

[4] 陈瑞 . 嵌入式 C 语言代码优化的研究 [J]. 科技信息 ,2009(13):54-55. DOI:10.3969/j.issn.1001-9960.2009.13.034.

[5] 周小丽，李俊峰 . 嵌入式系统编程中的代码优化 [J]. 山东电子 ,2004(2):60-62. DOI:10.3969/j.issn.1672-9528.2004.02.038.

[6] 周小丽，李俊峰 . 嵌入式系统编程中的代码优化 [J]. 电脑开发与应用 ,2004,17(1):4-6. DOI:10.3969/j.issn.1003-5850.2004.01.008.

[7] 韩芝萍 . 浅谈 C 代码优化与运用 [J]. 农业网络信息 ,2009(2):130-131. DOI:10.3969/j.issn.1672-6251.2009.02.042.

[8] 徐玉莲 . 基于嵌入式系统的 C 语言编程软件设计研究 [J]. 微型电脑应用 ,2019,35(10):113-114131.

[9] 曹悦，王红敏，张健雨 . 嵌入式软件的代码优化技术研究 [J/OL]. 中文科技期刊数据库（全文版）工程技术 ,2024,(11):229-232[2026-08-26].http://qikan.cqvip.com/Qikan/Article/Detail?id=1000004231345.

[10] 明芳 . 嵌入式系统程序代码优化方法 [J]. 计算机与数字工程 ,2006,34(10):189-192.

 Copyright © This Work is Licensed under A Commons Attribution-Non Commercial 4.0 International License.