Application Note

异构 SOC 上的主域恢复

Vinit Patel, Sudheer Kumar

摘要

使用异构多核处理器的汽车和工业系统需要高可靠性以及自动故障检测和恢复功能。TI TDA4x 片上系统 (SoC) 包含多个处理内核，包括 ARM Cortex-A72、Cortex-R5F 和 C7x DSP 内核，这些内核必须能够在无需手动干预的情况下连续运行。本应用手册介绍了如何使用 TDA4x 平台上的处理器间通信 (IPC) 驱动程序和低功耗管理 (LPM) 驱动程序来实现内核间信号监控系统。该解决方案通过乒乓协议实时监控所有远程内核，在可配置的时间窗口内进行自动崩溃检测，并通过仅 MCU 模式电源周期进行自主恢复。测试结果表明总恢复时间约为 1.2 秒，比整个系统电源周期快。本文提供了理论、实现细节和测试结果，使工程师能够在基于 TDA4x 的系统中实现类似的容错设计。

内容

商标

1 简介

TDA4x 是一款高性能异构多核片上系统 (SoC)，专为汽车和工业应用而设计。它集成了多个处理元件，包括：

- 用于高级操作系统执行的多个 ARM Cortex-A72 内核
- 多个用于实时控制任务的 ARM Cortex-R5F 内核
- 多个用于计算密集型算法的 C7x 数字信号处理器

在生产部署中，这些内核持续运行、并且必须保持高可用性。单个内核可能会由于软件错误、内存损坏、无限循环或硬件故障而无响应。如果没有自动检测和恢复机制，这种故障就需要手动干预，这在汽车和工业环境中是不可接受的。

本应用手册通过实现主域系统的信号监控和自动恢复来解决这些难题，其具有以下功能：

- 实时监控所有远程内核（RTOS 和 Linux）
- 通过 IPC 乒乓协议进行自动崩溃检测
- 通过 LPM 驱动程序进行自主恢复
- 可配置的监控间隔和重试机制
- 详细的时序曲线绘制，用于性能分析

此设计利用 SOC 的常开型 MCU 域，该域在仅 MCU 模式转换期间保持供电状态。MCU1_0 内核用作检测信号监控器，持续检查所有其他内核的响应情况。检测到内核故障时，系统会自动对主域实施电源周期，同时保持 MCU 域状态，从而实现快速恢复，而无需重新启动整个系统。

本文档提供：

- 详细的系统架构和设计原理
- 测试结果与时序测量

目标受众：本应用手册适用于使用 Processor SDK RTOS 在 jasinto 平台上开发容错系统的嵌入式软件工程师。

必要条件：

- 熟悉 TDA4x 架构
- 了解 IPC 和 remoteproc 框架
- 在 TI 处理器上使用 FreeRTOS 和 Linux 的经验

2 系统架构

2.1 内核配置

TDA4x SoC 包含多个组织成集群的处理内核。表 2-1 列出了此实现中使用的内核配置。

表 2-1. 内核配置

内核	类型	操作系统	角色
MCU1_0	R5F	FreeRTOS	LPM 驱动程序 + 检测信号监控器
MCU1_1	R5F	FreeRTOS	此配置中未使用
MCU2_0	R5F	FreeRTOS	远程内核 (受监控)
MCU2_1	R5F	FreeRTOS	远程内核 (受监控)
MCU3_0	R5F	FreeRTOS	远程内核 (受监控)
MCU3_1	R5F	FreeRTOS	远程内核 (受监控)
MPU1_0	A72	Linux	Linux 内核 (受监控)
C7x_1	DSP	FreeRTOS	DSP 内核 (受监控)

MCU1_0 内核被指定为检测信号监控器，因为该内核位于 MCU 域中，而 MCU 域在仅 MCU 模式期间保持通电状态。这使监控器能够承受主域电源周期并协调恢复。

2.2 电源域架构

TDA4x SOC 将子系统组织为三个电源域，从而实现选择性电源控制。表 2-2 列出了这些域。

表 2-2. 电源域

域	内容
WKUP 域	常开逻辑、唤醒控制器、WKUP 外设
MCU 域	MCU1_0、MCU1_1、MCU 外设、MCU SRAM
MAIN 域	A72 集群、MCU2-4 R5F 内核、C7x DSP、DDR、大多数 I/O

LPM 驱动程序支持两种主要功耗模式：

1. **活动模式：**所有三个域均已通电。这是所有内核执行应用程序的正常运行状态。
2. **仅 MCU 模式：**仅 WKUP 和 MCU 域通电。主域完全断电，包括所有 A72 内核、主域 R5F 内核、C7x DSP 和 DDR 控制器。

该恢复机制利用以下功耗模式：

活动 -> 仅 MCU -> 活动

该序列在 MCU 域 (即我们的机制) 保持运行时，会对主域实施电源周期。TPS6594x PMIC 在 MCU1_0 的 I2C 控制下管理电压轨。

2.3 IPC 框架概述

处理器间通信 (IPC) 框架支持在内核之间传递消息。此实现使用 RPMessage API，它提供：

- 基于虚拟 IO 的共享内存传输

- 用于建立连接的命名服务端点

- 超时状态异步消息发送和接收

两种 IPC 服务用于检测信号监控：

1. **ti.ipc4.ping-pong（端点 13）：**用于 RTOS 内核通信。MCU1_0 监控器发送 ping 消息，并预期远程 RTOS 内核做出 pong 响应。
2. **rpmsg_chrdev（端点 14）：**用于 Linux 通信。对于用户空间应用程序，Linux 需要字符设备接口 (rpmsg_char) 而非内核 RPMsg 驱动程序。

双重服务方法是必要的，因为：

- Linux 动态地分配端点，需要初始握手

- RTOS 内核使用静态端点分配

3 检测信号监控设计

3.1 乒乓协议

检测信号监控使用简单的乒乓协议：

协议运行：

1. 监控器发送“ping N”消息，其中 N 是序列号
2. 远程内核以“pong N”进行响应
3. 如果超时前没有响应，则重试至 MAX_RETRIES
4. 如果所有重试均失败，则声明内核已损毁
5. 协议参数：

   时间间隔：监控周期间隔 500ms

   超时：每个 ping 响应等待 2000ms

   最大重试次数：在声明失败前应尝试三次

3.2 双任务架构

两个单独的 FreeRTOS 任务负责监控不同的内核类型：

发送器任务（ti.ipc4.ping-pong 服务）：

- 监控 RTOS 内核：mcu2_0、mcu2_1、mcu3_0、mcu3_1、c7x_1
- 使用端点 13
- MCU 启动通信：发送 ping，接收 pong
- 在每个监控周期中迭代所有 RTOS 内核

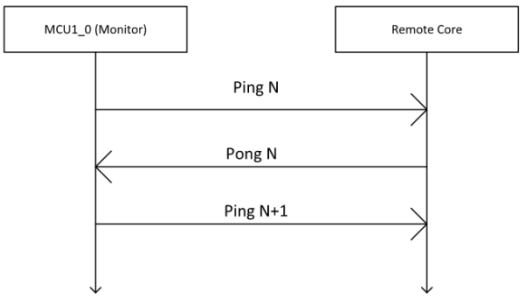

图 3-1. SenderTask 机制

响应者任务（rpmsg_chrdev 服务）：

- 监控 Linux 内核：mpu1_0
- 使用端点 14
- 等待初始“Linux 就绪”消息以采集动态端点
- 然后 MCU 启动：发送 ping，Linux 以 pong 进行响应
- 使用第一条虚拟消息的原因是 Linux 动态地分配端点，并且 MCU1_0（监控器）通过虚拟消息获取 Linux 的远程端点进行通信。

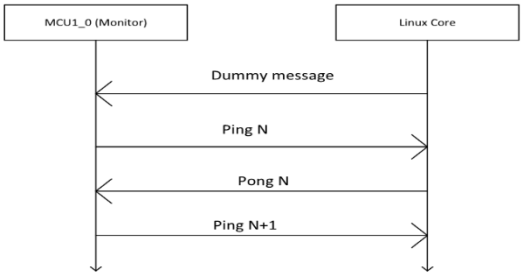

图 3-2. ResponderTask 机制

表 3-1 显示了每个内核的监控分配。

表 3-1. 监控分配

内核	服务	终点	监控方
mpu1_0 (Linux)	rpmsg_chrdev	14	响应者任务
mcu2_0	ti.ipc4.ping-pong	13	发送方任务
mcu2_1	ti.ipc4.ping-pong	13	发送方任务
mcu3_0	ti.ipc4.ping-pong	13	发送方任务
mcu3_1	ti.ipc4.ping-pong	13	发送方任务
c7x_1	ti.ipc4.ping-pong	13	发送方任务

分为两个任务具有以下几个优点：

- 具有不同时序要求的独立监控回路

- 隔离特定于 Linux 的握手逻辑

- 并行监控能力

3.3 崩溃检测逻辑

崩溃检测逻辑会为每个受监控的内核保持状态：

状态转换：

- **未响应 -> 活动：**当初始化或恢复后首次成功收到 Pong 时。记录为“恢复：内核 X 现在为活动状态！”
- **活动 -> 已崩溃：**在所有重试次数用完后 ping 失败时。记录为“提醒：内核 X 未响应！”这会触发崩溃标志。

机制简介：

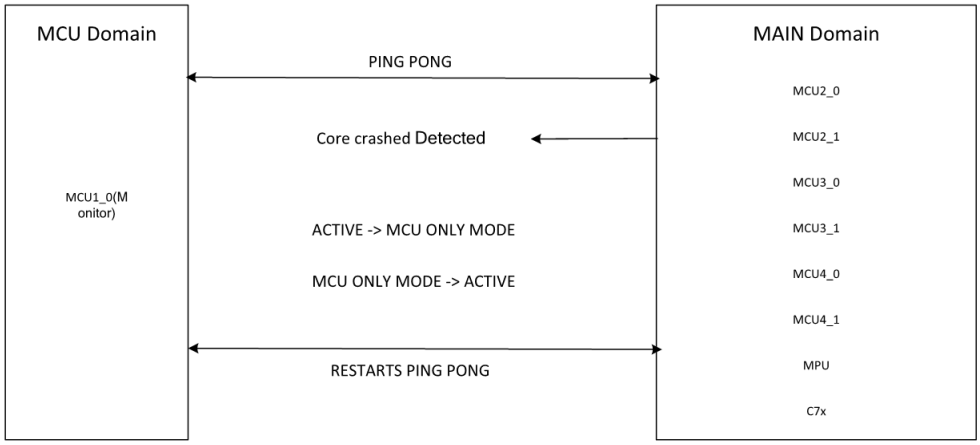

图 3-3. 简要工作流程

4 恢复机制

4.1 电源状态转换

当检测到崩溃时，恢复机制会执行以下序列：

1. 指示要退出的 IPC 任务
2. 等待所有任务通过信标退出
3. 取消初始化 IPC 框架
4. 将资源表保存到隐藏位置
5. 转换活动 -> 仅 MCU（关闭主域电源）
6. 转换仅 MCU -> 活动（打开主域电源）
7. 从隐藏位置恢复资源表
8. 重新初始化 IPC 框架
9. 启动远程内核
10. 恢复监控机制

功率转换在 LPM 驱动程序中实现：

Lpm_activeToMcuSwitch()：

- 向主域发出软件复位
- 启用主域深度睡眠隔离
- PMIC 状态更改：活动 -> 仅 MCU 通过 I2C 连接

Lpm_mcuToActiveSwitch()：

- PMIC 状态更改：仅 MCU -> 活动 通过 I2C 连接
- 禁用主域深度睡眠隔离
- 启用 WKUPMCU2MAIN 和 MAIN2WKUPMCU 桥
- 重新初始化电路板配置
- 恢复 DDR 控制器、PLL 和时钟

5 实现详情

5.1 配置参数

检测信号监控参数在 lpm_ipc.c 中定义：

表 5-1. 配置参数

参数	值	说明
HEARTBEAT_INTERVAL_MS	500	监控周期之间的时间 (ms)
HEARTBEAT_TIMEOUT_MS	2000	每次 ping 响应的超时时间 (ms)
HEARTBEAT_MAX_RETRIES	3	在声明失败前重试

5.2 参数调优指南：

HEARTBEAT_INTERVAL_MS 较低：

+ 更快地检测内核故障

- 更高的 IPC 开销和 CPU 利用率

HEARTBEAT_INTERVAL_MS 较高：

+ 更低的系统开销

- 更慢的崩溃检测

HEARTBEAT_TIMEOUT_MS 较低：

+ 每次重试时检测故障的速度更快

- 如果内核处于重负载下，会导致误报

HEARTBEAT_MAX_RETRIES 较高：

+ 更好地抵御瞬态通信延迟

- 更长的检测实际崩溃的时间

5.3 Linux 端的 rpmsg_char 实现

检测信号监控系统需要 Linux 用户空间应用程序来处理与 MCU1_0 监控器的通信。此实现使用 TI rpmsg_char 库和框架将用户空间应用程序与内核 RPMsg 子系统桥接在一起。

ti-rpmsg-char 库为 Linux 上的 RPMsg 通信提供用户空间 API。这包括：

- **核心库：**libti_rpmsg_char.so.0.6.10 - 处理端点管理和内核接口

- **简单应用程序：**rpmsg_char_simple - 用于信号通信到 Linux 内核

6 测试结果和性能

6.1 时序曲线

时序曲线系统在恢复过程中的关键点采集时间戳。表 6-1 显示实际测试运行的测量值。

表 6-1. 时间曲线

相位	持续时间 (usec)	时长 (ms)
崩溃检测 -> 启动仅 MCU 准备	34 - 38	约 0.035
活动 -> 仅 MCU 切换	7011 - 7044	约 7.0
仅 MCU -> 活动切换	150711 - 150752	约 150.7
Linux A72 内核启动	1069435	约 1069.4
总恢复时间		约 1227

总恢复时间约为 1227ms，而完整电源周期启动时间约为 1333ms，在保持所有 MCU 域状态的同时可节省约 105ms。

6.2 恢复验证

检测信号监控系统通过多种崩溃场景进行了测试：

测试用例 1：RTOS 内核崩溃 (c7x_1)

- 通过停止 C7x DSP 来模拟
- 检测时间：3 个重试周期
- 恢复：成功，所有内核都恢复到活动状态

测试用例 2：R5F 内核崩溃 (mcu2_0)

- 通过停止 MCU2_0 内核来模拟
- 检测时间：3 个重试周期
- 恢复：成功，所有内核都恢复到活动状态

测试用例 3：Linux 内核崩溃 (mpu1_0)

- 通过终止 rpmsg_char_simpled 应用程序或停止 mpu1_0 内核来模拟
- 检测时间：3 个重试周期
- 恢复：成功，Linux 重新启动并重新连接，所有内核都恢复到活动状态

7 总结

本应用手册介绍了 TDA4x 平台的内核间检测信号监控和自动恢复的完整实现。主要贡献包括：

1. 双任务监控架构：RTOS 和 Linux 内核的单独任务可实现独立监控，并为每个环境提供适当的超时行为。
2. 乒乓协议：简单的请求响应协议通过可配置的参数提供可靠的内核运行状况验证。
3. 基于信标的同步：正确的任务同步验证了在没有竞态条件的情况下实现可靠的 IPC 关断。
4. 仅 MCU 模式恢复：利用 SOC 的电源域架构，无需重新启动整个系统即可实现自主恢复。
5. 时序曲线：详细的时序测量显示恢复性能约为 1.2 秒。

此实现为具有高可用性和自动恢复等基本要求的容错汽车和工业系统奠定了基础。

未来的增强功能包括：

- 选择性内核恢复 (仅重新启动崩溃的内核，而不是重启整个主域的电源周期)

8 参考资料

1. 德州仪器 (TI)，*仅 MCU 模式简介*，用户指南。
2. 德州仪器 (TI)，*如何启用信号监控和主域内核的自主恢复*，常见问题解答。
3. 德州仪器 (TI)，*如何在 TDA4VM 上运行和验证仅 MCU 模式*，常见问题解答。
4. 德州仪器 (TI)，*如何在 TDAVH 上运行和验证仅 MCU 模式*，常见问题解答。
5. 德州仪器 (TI)，*Linux rpmsg_char 驱动程序指南*，用户指南。

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月