Application Note

Recovery of Main Domain on Heterogeneous SOC

Vinit Patel, Sudheer Kumar

ABSTRACT

Automotive and industrial systems using heterogeneous multi-core processors require high reliability with automatic fault detection and recovery capabilities. The TI TDA4x System-on-Chip (SoC) contains multiple processing cores including ARM Cortex-A72, Cortex-R5F, and C7x DSP cores that must operate continuously without manual intervention. This application note describes the implementation of an inter-core heartbeat monitoring system using the Inter-Processor Communication (IPC) driver and Low Power Management (LPM) driver on the TDA4x platform. The solution provides real-time monitoring of all remote cores via a ping-pong protocol, automatic crash detection within configurable time windows, and autonomous recovery through MCU Only Mode power cycling. Test results demonstrate total recovery time of approximately 1.2 seconds, which is faster than a full system power cycle. This document provides the theory, implementation details, and test results to enable engineers to implement similar fault-tolerant designs in TDA4x based systems.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

Copyright © 2026 Texas Instruments Incorporated

1 Introduction

The TDA4x is a high-performance heterogeneous multi-core System-on-Chip (SoC) designed for automotive and industrial applications. This integrates multiple processing elements including:

- Multiple ARM Cortex-A72 cores for high-level operating system execution

- Many ARM Cortex-R5F cores for real-time control tasks

- Several C7x Digital Signal Processors for compute-intensive algorithms

In production deployments, these cores operate continuously and must maintain high availability. Individual cores can become unresponsive due to software bugs, memory corruption, infinite loops, or hardware faults. Without automatic detection and recovery mechanisms, such failures require manual intervention, which is unacceptable in automotive and industrial environments.

This application note addresses these challenges by implementing a heartbeat monitoring and automatic recovery of main domain system with the following capabilities:

- Real-time monitoring of all remote cores (RTOS and Linux)

- Automatic crash detection via IPC ping-pong protocol

- Autonomous recovery through LPM driver

- Configurable monitoring intervals and retry mechanisms

- Detailed timing profiling for performance analysis

The design leverages the always-on MCU domain of the SOC, which remains powered during MCU Only Mode transitions. The MCU1_0 core serves as the heartbeat monitor, continuously checking the responsiveness of all other cores. When a core failure is detected, the system automatically power cycle the MAIN domain while preserving the MCU domain state, enabling rapid recovery without full system reboot.

This document provides:

- Detailed system architecture and design rationale

- Test results with timing measurements

Target Audience: This application note is intended for embedded software engineers developing fault-tolerant systems on the jasinto platform using the Processor SDK RTOS.

Prerequisites:

- Familiarity with TDA4x architecture

- Understanding of IPC and remoteproc frameworks

- Experience with FreeRTOS and Linux on TI processors

Copyright © 2026 Texas Instruments Incorporated

2 System Architecture

2.1 Core Configuration

The TDA4x SoC contains multiple processing cores organized into clusters. Table 2-1 lists the core configuration used in this implementation.

Table 2-1. Core Configuration

Core	Type	Operating System	Role
MCU1_0	R5F	FreeRTOS	LPM Driver + Heartbeat Monitor
MCU1_1	R5F	FreeRTOS	Not used in this configuration
MCU2_0	R5F	FreeRTOS	Remote core (monitored)
MCU2_1	R5F	FreeRTOS	Remote core (monitored)
MCU3_0	R5F	FreeRTOS	Remote core (monitored)
MCU3_1	R5F	FreeRTOS	Remote core (monitored)
MPU1_0	A72	Linux	Linux core (monitored)
C7x_1	DSP	FreeRTOS	DSP core (monitored)

The MCU1_0 core is designated as the heartbeat monitor because the core resides in the MCU domain, which remains powered during MCU Only Mode. This allows the monitor to survive MAIN domain power cycles and orchestrate recovery.

2.2 Power Domain Architecture

The TDA4x SOCs organizes subsystems into three power domains, enabling selective power control. Table 2-2 lists these domains.

Table 2-2. Power Domains

Domain	Contents
WKUP Domain	Always-on logic, wake-up controllers, WKUP peripherals
MCU Domain	MCU1_0, MCU1_1, MCU peripherals, MCU SRAM
MAIN Domain	A72 cluster, MCU2-4 R5F cores, C7x DSPs, DDR, most I/O

The LPM driver supports two primary power modes:

1. **ACTIVE Mode:** All three domains are powered. This is the normal operating state where all cores execute applications.
2. **MCU ONLY Mode:** Only WKUP and MCU domains are powered. The MAIN domain is completely powered off, including all A72 cores, MAIN domain R5F cores, C7x DSPs, and DDR controller.

The recovery mechanism leverages these power modes:

ACTIVE -> MCU ONLY -> ACTIVE

This sequence power cycles the MAIN domain while the MCU domain (Hence, our mechanism) remains operational. The TPS6594x PMIC manages the voltage rails under I2C control from MCU1_0.

Copyright © 2026 Texas Instruments Incorporated

2.3 IPC Framework Overview

The Inter-Processor Communication (IPC) framework enables message passing between cores. This implementation uses the RPMessage API, which provides:

- VirtIO-based transport over shared memory

- Named service endpoints for connection establishment

- Asynchronous message send and receive with timeout

Two IPC services are used for heartbeat monitoring:

1. **ti.ipc4.ping-pong (Endpoint 13):** Used for RTOS core communication. The MCU1_0 monitor sends ping messages and expects pong responses from remote RTOS cores.
2. **rpmsg_chrdev (Endpoint 14):** Used for Linux communication. Linux requires a character device interface (rpmsg_char) rather than the kernel RPMsg driver for user-space applications.

The dual-service approach is necessary because:

- Linux dynamically assigns endpoints, requiring initial handshake

- RTOS cores use static endpoint assignments

3 Heartbeat Monitoring Design

3.1 Ping-Pong Protocol

The heartbeat monitoring uses a simple ping-pong protocol:

Protocol Operation:

1. Monitor sends "ping N" message where N is a sequence number
2. Remote core responds with "pong N"
3. If no response within timeout, retry up to MAX_RETRIES
4. If all retries fail, declare the core as crashed
5. Protocol Parameters:

   Interval: 500ms between monitoring cycles

   Timeout: 2000ms wait for each ping response

   Max Retries: Three attempts before declaring failure

3.2 Dual-Task Architecture

Two separate FreeRTOS tasks handle monitoring different core types:

Sender Task (ti.ipc4.ping-pong service):

- Monitors RTOS cores: mcu2_0, mcu2_1, mcu3_0, mcu3_1, c7x_1
- Uses endpoint 13
- MCU initiates communication: sends ping, receives pong
- Iterates through all RTOS cores in each monitoring cycle

Copyright © 2026 Texas Instruments Incorporated

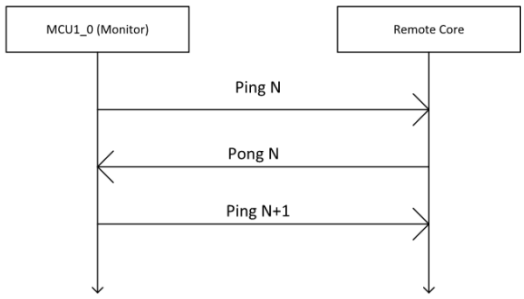

Figure 3-1. SenderTask Mechanism

Responder Task (rpmsg_chrdev service):

- Monitors Linux core: mpu1_0
- Uses endpoint 14
- Waits for initial "Linux ready" message to capture dynamic endpoint
- Then MCU initiates: sends ping, Linux responds with pong
- The reason behind using the first dummy message is that Linux assigns the end point dynamically and with the dummy message the MCU1_0(Monitor) gets the remote endpoint of Linux to communicate with.

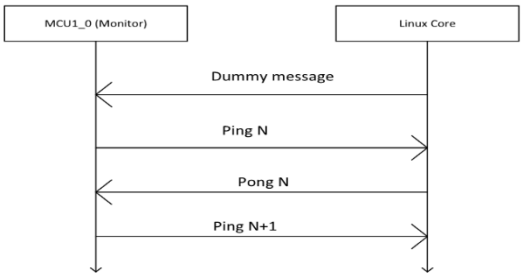

Figure 3-2. ResponderTask Mechanism

Table 3-1 shows the monitoring assignment for each core.

Table 3-1. Monitoring Assignment

Core	Service	Endpoint	Monitored By
mpu1_0 (Linux)	rpmsg_chrdev	14	Responder Task
mcu2_0	ti.ipc4.ping-pong	13	Sender Task
mcu2_1	ti.ipc4.ping-pong	13	Sender Task
mcu3_0	ti.ipc4.ping-pong	13	Sender Task
mcu3_1	ti.ipc4.ping-pong	13	Sender Task
c7x_1	ti.ipc4.ping-pong	13	Sender Task

The separation into two tasks provides several benefits:

- Independent monitoring loops with different timing requirements

- Isolation of Linux-specific handshake logic

- Parallel monitoring capability

Copyright © 2026 Texas Instruments Incorporated

3.3 Crash Detection Logic

The crash detection logic maintains state for each monitored core:

State Transitions:

- **NOT_RESPONDING -> ALIVE:** When first successful pong is received after initialization or recovery. Logged as "RECOVERY: Core X is now ALIVE!"
- **ALIVE -> CRASHED:** When ping fails after all retries are exhausted. Logged as "ALERT: Core X is NOT RESPONDING!" This triggers the crash flag.

Mechanism in brief:

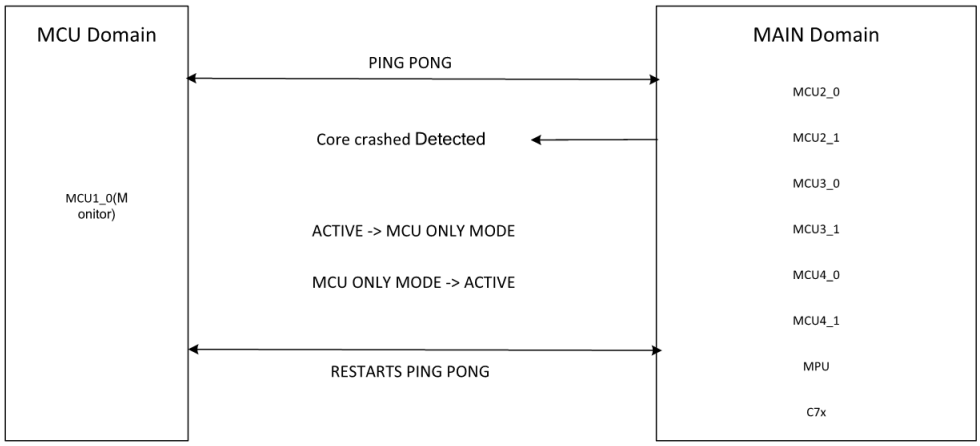

Figure 3-3. Brief Workflow

Copyright © 2026 Texas Instruments Incorporated

4 Recovery Mechanism

4.1 Power State Transitions

When a crash is detected, the recovery mechanism executes the following sequence:

1. Signal IPC tasks to exit
2. Wait for all tasks to exit via semaphores
3. Deinitialize IPC framework
4. Save resource table to shadow location
5. Transition ACTIVE -> MCU ONLY (power off MAIN domain)
6. Transition MCU ONLY -> ACTIVE (power on MAIN domain)
7. Restore resource table from shadow location
8. Reinitialize IPC framework
9. Boot remote cores
10. Resume mechanism of monitoring

The power transitions are implemented in the LPM driver:

Lpm_activeToMcuSwitch():

- Issue software reset to MAIN domain
- Enable MAIN domain deep sleep isolation
- PMIC state change: ACTIVE -> MCU ONLY via I2C

Lpm_mcuToActiveSwitch():

- PMIC state change: MCU ONLY -> ACTIVE via I2C
- Disable MAIN domain deep sleep isolation
- Enable WKUPMCU2MAIN and MAIN2WKUPMCU bridges
- Re-initialize board configuration
- Restore DDR controller, PLLs, and clocks

Copyright © 2026 Texas Instruments Incorporated

5 Implementation Details

5.1 Configuration Parameters

The heartbeat monitoring parameters are defined in lpm_ipc.c:

Table 5-1. Configuration Parameters

Parameter	Value	Description
HEARTBEAT_INTERVAL_MS	500	Time between monitoring cycles (ms)
HEARTBEAT_TIMEOUT_MS	2000	Timeout for each ping response (ms)
HEARTBEAT_MAX_RETRIES	3	Retries before declaring failure

5.2 Parameter Tuning Guidance:

Lower HEARTBEAT_INTERVAL_MS:

+ Faster detection of core failures

- Higher IPC overhead and CPU utilization

Higher HEARTBEAT_INTERVAL_MS:

+ Lower system overhead

- Slower crash detection

Lower HEARTBEAT_TIMEOUT_MS:

+ Faster failure detection per retry

- Can cause false positives if cores are under heavy load

Higher HEARTBEAT_MAX_RETRIES:

+ More resilient to transient communication delays

- Longer time to detect actual crashes

5.3 Linux-Side rpmsg_char Implementation

The heartbeat monitoring system requires a Linux user-space application to handle communication with the MCU1_0 monitor. The implementation uses the TI rpmsg_char library and framework to bridge user-space applications with the kernel RPMsg subsystem.

The ti-rpmsg-char library provides a user-space API for RPMsg communication on Linux. This consists of:

- **Core Library:** libti_rpmsg_char.so.0.6.10 - Handles endpoint management and kernel interface

- **Simple Application:** rpmsg_char_simple - Used it for heartbeat communication to the linux core

Copyright © 2026 Texas Instruments Incorporated

6 Test Results and Performance

6.1 Timing Profile

The timing profiling system captures timestamps at key points during the recovery process. Table 6-1 shows measured values from actual test runs.

Table 6-1. Time Profile

Phase	Duration (usec)	Duration (ms)
Crash Detection -> Start MCU ONLY Prep	34 - 38	Approximately 0.035
ACTIVE -> MCU ONLY Transition	7011 - 7044	Approximately 7.0
MCU ONLY -> ACTIVE Transition	150711 - 150752	Approximately 150.7
Linux A72 Core Boot	1069435	Approximately 1069.4
TOTAL RECOVERY TIME		Approximately 1227

The total recovery time of approximately 1227ms compares favorably to a full power cycle boot time of approximately 1333ms, providing a savings of about 105ms while maintaining all MCU domain state.

6.2 Recovery Verification

The heartbeat monitoring system was tested with multiple crash scenarios:

Test Case 1: RTOS Core Crash (c7x_1)

- Simulated by halting the C7x DSP

- Detection time: 3 retry cycles

- Recovery: Successful, all cores restored to ALIVE state

Test Case 2: R5F Core Crash (mcu2_0)

- Simulated halting MCU2_0 core

- Detection time: 3 retry cycles

- Recovery: Successful, all cores restored to ALIVE state

Test Case 3: Linux Core Crash (mpu1_0)

- Simulated by killing the rpmsg_char_simple application or by halting the mpu1_0 core

- Detection time: 3 retry cycles

- Recovery: Successful, Linux rebooted and reconnected, all cores restored to ALIVE state

Copyright © 2026 Texas Instruments Incorporated

7 Summary

This application note presented a complete implementation of inter-core heartbeat monitoring and automatic recovery for the TDA4x platforms. The key contributions include:

1. Dual-Task Monitoring Architecture: Separate tasks for RTOS and Linux cores enable independent monitoring with appropriate timeout behaviors for each environment.
2. Ping-Pong Protocol: Simple request-response protocol provides reliable core health verification with configurable parameters.
3. Semaphore-Based Synchronization: Proper task synchronization verifies reliable IPC shutdown without race conditions.
4. MCU Only Mode Recovery: Leveraging the SOC's power domain architecture enables autonomous recovery without full system reboot.
5. Timing Profile: Detailed timing measurements demonstrate recovery performance of approximately 1.2 seconds.

The implementation provides a foundation for fault-tolerant automotive and industrial systems where high availability and automatic recovery are essential requirements.

Future enhancements can include:

- Selective core recovery (restart only crashed core without full MAIN domain power cycle)

8 References

1. Texas Instruments, *MCU only mode Introduction*, user's guide.
2. Texas Instruments, *How to enable heartbeat monitoring and autonomous recovery of main domain cores*, FAQ.
3. Texas Instruments, *How to run and validate MCU only mode on TDA4VM*, FAQ.
4. Texas Instruments, *How to run and validate MCU only mode on TDAVH*, FAQ.
5. Texas Instruments, *Linux rpmsg_char driver guide*, user's guide.

Copyright © 2026 Texas Instruments Incorporated

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Copyright © 2026, Texas Instruments Incorporated

Last updated 10/2025