Application Note

了解 TDA4VM 或 DRA829 存储器以获得出色性能

Karan Saxena, Kip Broadhurst, Junbok You, Richard Woodruff, and Ganesh Chelliah

摘要

Jacinto 7 系列器件支持多个内部存储器，可为大量汽车和工业应用实现出色的系统延迟、带宽和功能安全特性。器件特定数据表和技术参考手册 (TRM) 可提供器件中不同类型存储器及其预期用途的详细概述。

本应用手册专门针对 TDA4VM，简要概述了不同的存储器类型、典型用例以及软件开发套件 (SDK) 注意事项。

TDA4VM	
数据表	https://www.ti.com/cn/lit/gpn/tda4vm
TRM	https://www.ti.com/cn/lit/zip/spruil1
SDK	https://www.ti.com.cn/tool/cn/PROCESSOR-SDK-J721E

有关本应用手册未涵盖内容的详细信息，请参阅*适用于 ADAS 和自动驾驶汽车的 TDA4VM Jacinto™ 处理器器件修订版 1.0 和 1.1 数据表* 和 *DRA829/TDA4VM 技术参考手册*。如果需要进一步说明，请在 TI 处理器的 E2E 论坛上发帖。

内容

商标

1 TDA4VM 上不同类型的存储器

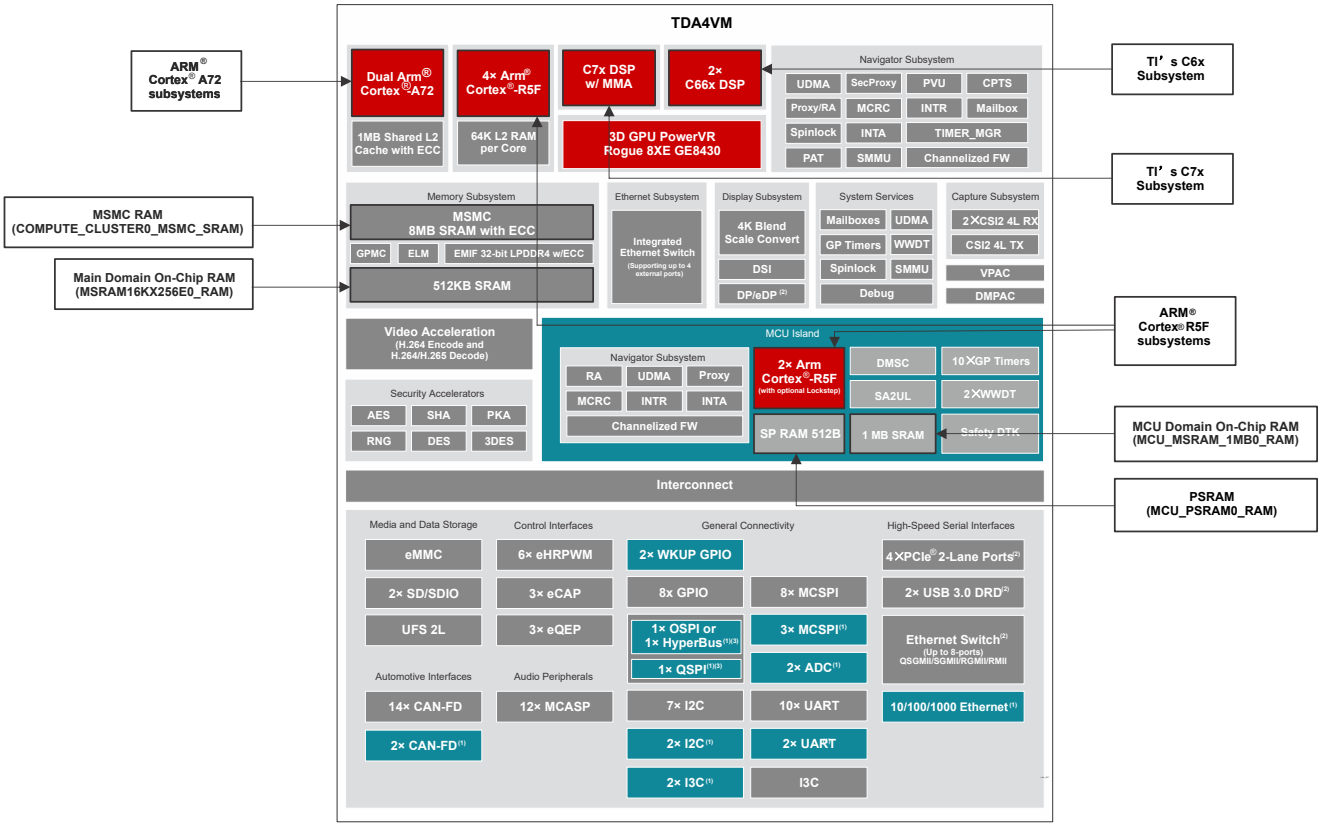

图 1-1. 器件框图

1. 该接口位于 MCU 岛上，但整个系统都可以访问该接口。
2. DP、SGMII、USB3.0 和 PCIE[3:0] 共用总共 12 个串行器/解串器通道。
3. 2 个同步闪存接口，配置为 OSPI0 和 OSPI1 或 HyperBus™ 和 OSPI1。

本文档介绍了 TDA4VM 上的以下存储器和子系统：

1. PSROM
2. PSRAM
3. MSMC RAM
4. MSRAM
5. Arm® Cortex® A72 子系统中的存储器
6. Arm Cortex R5F 子系统中的存储器
7. TI C6x 子系统中的存储器
8. TI C7x 子系统中的存储器
9. DDR 子系统

2 存储器概述和预期用途

本节详细介绍了器件上不同类型的存储器及其典型用例。这些信息对于设计终端系统的存储器要求至关重要。

2.1 PSROM

PSROM 模块提供了一个可用于访问 ROM 的存储器映射区域。该模块没有寄存器，并将总线接口地址、控制和读取数据信号映射到 ROM 的地址、控制和读取数据。有关更多信息，请参阅 *DRA829/TDA4VM 技术参考手册* 中的 *器件的存储器映射* 一章。

2.1.1 典型使用案例

PSROM 用于引导 ROM。引导 ROM 是一种非易失性存储器，在器件启动期间用于加载客户定义的映像，例如次级引导加载程序。该器件具有两个协同工作的 ROM 代码（在 TDA4VM 中）：MCU ROM 代码和 DMSC ROM 代码。当 MCU 处理器从复位状态释放时，引导 ROM 代码执行。此复位是从 DMSC 子系统生成的。ROM 仅在器件初始加电或从低功耗模式唤醒后执行。

2.2 PSRAM

PSRAM 块提供一个存储器映射区域，该区域可被实例化系统和系统中的其他主控器用来存储不同类型的信息。该区域用作标准的字节可访问静态 RAM。PSRAM 是一个连接到 RAM 的 vbusp 总线接口。它没有控制寄存器，并将总线接口地址、控制和数据信号映射到 RAM 的地址、控制和数据。VBUSP 协议是一种单发（即挂起）的强顺序协议，它强调简单性而不是可能的最高性能。

大小为 512 字节的 MCU_PSRAM0_RAM 将能够承受任何复位（MCU_PORz、PORz 或热复位），只要 MCU 域通过复位供电即可。

2.2.1 典型使用案例

- 暂存 RAM
  - 用于跨芯片复位存储调试和上下文信息
  - 在（热/供电）复位后，可以分析导致复位的事件，并调试软件。

2.3 MSMC RAM

多核共享存储器控制器 (MSMC) 构成计算集群 (COMPUTE_CLUSTER0) 的核心，可提供与所有连接的处理元件和系统其余部分之间的高带宽资源访问。MSMC 用作计算集群的数据移动主干。

MSMC 子系统支持片上 MSMC SRAM，在 TDA4VM 上，该 SRAM 为带有 ECC 的 8MB（4 组 x 2MB）SRAM。该存储器为：

- 共享相干 2 级/3 级存储器映射 SRAM
- 共享相干 3 级高速缓存

MSMC SRAM 可配置为 SRAM 或 L3 高速缓存，也可配置为二者的组合。该存储器可用于提高特定模块的性能。

要了解如何将 MSMC 拆分为高速缓存和 SRAM，请参阅节 4.4.2

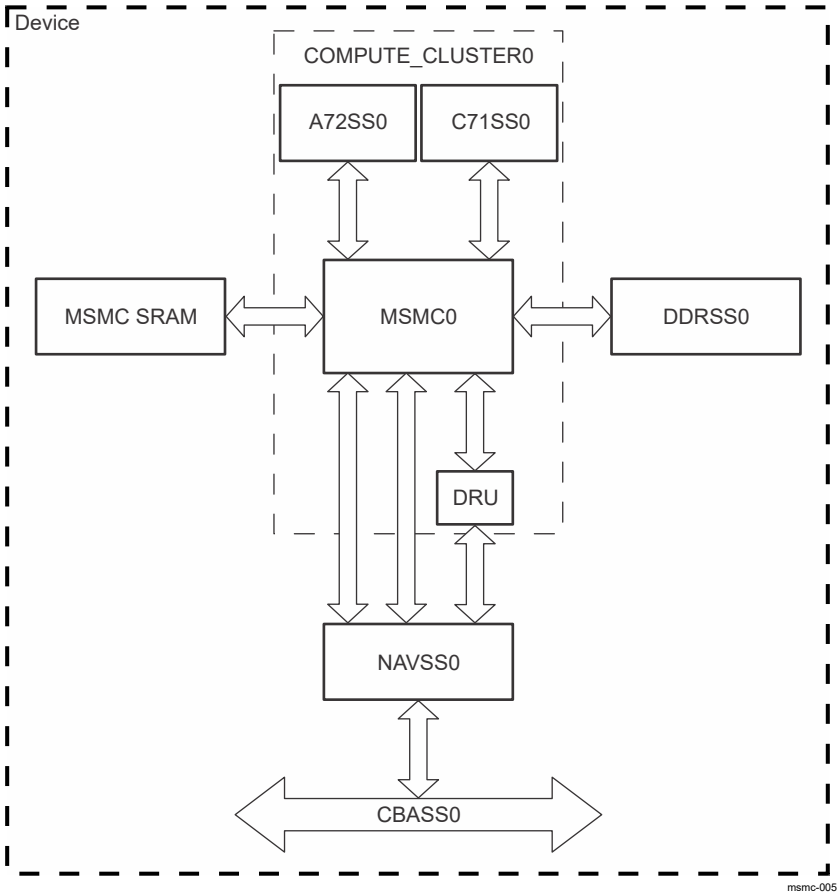

图 2-1. MSMC 概述

2.3.1 典型使用案例

- 计算集群的 L3 高速缓存（A72 和 C7x）
- 用于运行深度学习算法的 SRAM C7x

2.3.2 相关链接

有关更多信息，请参阅 *DRA829/TDA4VM 技术参考手册* 中的*多核共享存储器控制器 (MSMC)* 一章。

2.4 MSRAM

MSRAM 块提供一个存储器映射区域，此区域可用于存储不同类型的信息。它使用较高的事务处理问题速率总线接口，完全支持多个未完成的事务以及同时完成读取和写入事务。更多详细信息，请参阅器件特定 TRM 的*导航器子系统 (NAVSS)* 一章

器件上有两个 MSRAM：一个在主域中，大小为 512KB (*MSRAM16KX256E0_RAM*)；另一个在 MCU 域中，大小为 1MB (*MCU_MSRAM_1MB0_RAM*)。

2.4.1 典型使用案例

- 主域片上 SRAM 或主 OCRAM (*MSRAM16KX256E0_RAM*)
  - 提供额外的片上 ECC 保护 SRAM 以用于任何用途
- MCU 域片上 RAM 或 MCU OCRAM (*MCU_MSRAM_1MB0_RAM*)
  - 提供 RAM 用于程序/数据存储
  - R5F 和 DMSC/SMS 以及外部主器件可访问的本地 RAM（如果在其启动程序防火墙中已启用）
  - ROM 代码在引导期间以及加载引导映像时使用的片上 RAM 存储器

2.5 ARM Cortex A72 子系统

该器件实现了一个集成在计算集群内部以及其他模块中的双核 Arm Cortex-A72 MPU。Cortex-A72 内核是通用处理器，可用于运行客户应用程序。

2.5.1 L1/L2 高速缓存内存

- 每个处理器具有 48KB L1 指令高速缓存，具有奇偶校验保护
- 每个处理器具有 32KB L1 数据高速缓存，具有 ECC 保护
- 具备 ECC 保护的 1MB 共享 L2 高速缓存

2.5.2 L3 存储器

通过 MSMC 提供可选的 L3 高速缓存，请参阅节 2.3。

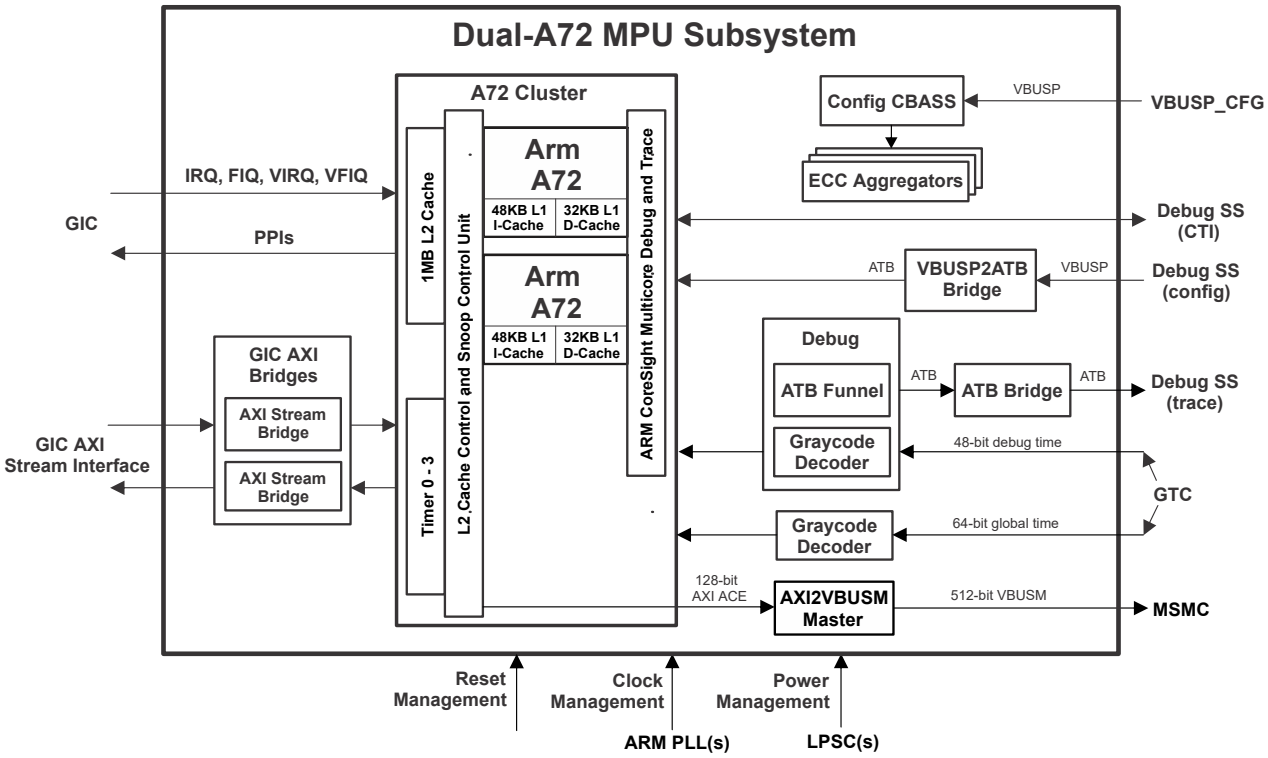

图 2-2. 双 A72 MPU 子系统

2.5.3 相关链接

- Arm Cortex-A72 主页：Cortex-A72 – Arm®
- Arm Cortex-A72 MPCore 处理器技术参考手册 r0p3：https://developer.arm.com/documentation/100095/0003
- 适用于 ARMv8-A 的 Arm Cortex-A 系列编程人员指南：https://developer.arm.com/documentation/den0024/a/
- 有关更多信息，请参阅 *DRA829/TDA4VM 技术参考手册* 中的*双 A72 MPU 子系统*一章。

2.6 ARM Cortex R5F 子系统

R5FSS 是 Arm Cortex-R5F 处理器的双核实现，配置为进行分离/锁定操作。它还包括附带的存储器（L1 高速缓存和紧密耦合存储器）、标准 Arm CoreSight™ 调试和布线架构、集成式矢量中断管理器 (VIM)、ECC 聚合器以及支持协议转换和地址转换的各种其他模块，以便于集成到 SoC。

2.6.1 L1 存储器系统

高速缓存和紧密耦合存储器 (TCM) 都是到 R5F 的 L1 存储器。

2.6.2 高速缓存

R5F 具有 Harvard 高速缓存架构，这意味着它具有独立的指令和数据高速缓存。

- 16KB 指令高速缓存
  - 4x4KB 路
  - 每 64 位受 SECDED ECC 保护
- 16KB 数据高速缓存
  - 4x4KB 路
  - 每 32 位受 SECDED ECC 保护

2.6.3 紧耦合存储器 (TCM)

集群中每个 CPU 有 64KB 的紧密耦合存储器。TCM 产生的单周期延迟，与在高速缓存中登录事务相同。

- 每 32 位受 SECDED ECC 保护
- 可从系统读取/写入
- 拆分为 A 组和 B 组（B 进一步拆分为 B0 和 B1 交错组）
  - 32KB TCMA (ATCM)
  - 16KB TCMB0 (B0TCM)
  - 16KB TCMB1 (B1TCM)

2.6.4 典型用例

为了获得出色性能，可将局部特性较差的热数据和代码放置在 TCM 中。TCM 有时会加载一些启动代码以初始化 R5F 存储器保护单元 (MPU)，从而进一步执行 DDR 之外的代码。由于某些应用程序无法很好地缓存，有两个 TCM 接口允许连接到*紧密耦合存储器*（ATCM 和 BTCM）的可配置存储器块。ATCM 通常用于代码，而双端口 BTCM 用于数据。

- ATCM 通常保存必须高速访问的中断或异常代码，而不会因缓存未命中而产生任何潜在延迟。
- BTCM 通常保存一个数据块以进行密集处理，例如音频或视频处理。BTCM 采用双端口以增加带宽。
- TCM 位于处理器外部。这在针对性能、功耗和 RAM 类型优化 TCM 子系统方面提供了灵活性。

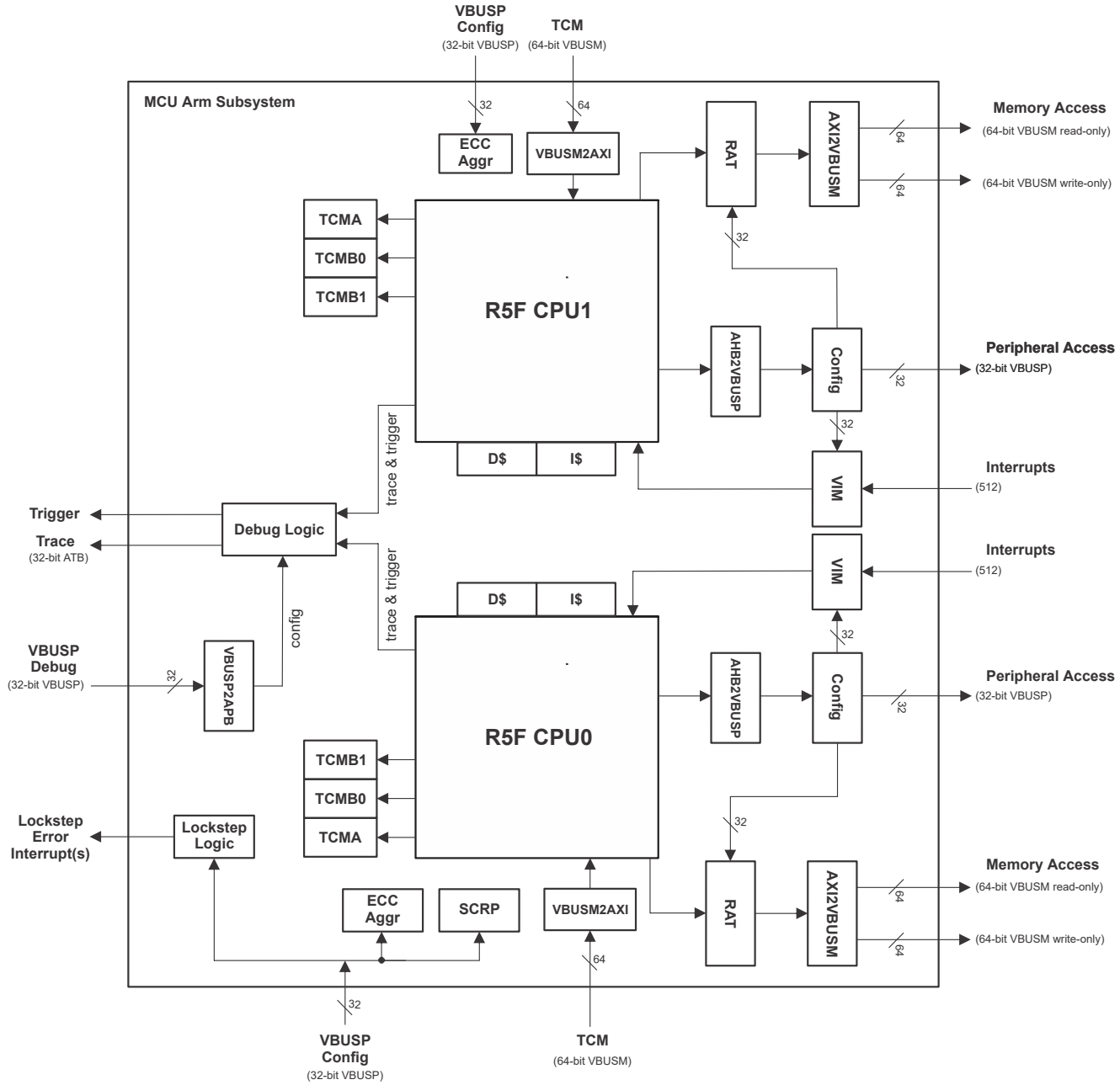

图 2-3. R5FSS 方框图

2.6.5 相关链接

- Arm Cortex-R5F 主页：Cortex-R5 – ARM
- Cortex-R5 技术参考手册 r1p2：https://developer.arm.com/documentation/ddi0460/d
- Arm Cortex-R 系列编程人员指南：https://developer.arm.com/documentation/den0042/a/Coding-for-Cortex-R-Processors
- 有关更多信息，请参阅 *DRA829/TDA4VM 技术参考手册* 中的*双 R5F MCU 子系统*一章。

2.7 TI 的 C6x 子系统

C66x 子系统基于 TI 的标准 TMS320C66x DSP CorePac 模块。它包含子系统逻辑，可简化 C66x CorePac 与 SoC 的集成，同时更大程度地重复使用以前器件中的软件。

2.7.1 存储器布局

- 程序存储器控制器 (PMC)：32KB L1 程序存储器 (L1P)，配置为 32 字节行高速缓存，2KB 页
- 数据存储器控制器 (DMC)：32KB L1 数据存储器，可配置为高速缓存和/或 SRAM
- 统一存储器控制器 (UMC)：288KB L2 存储器
  - 256KB 可配置为高速缓存或 SRAM
  - 32KB 始终为 SRAM

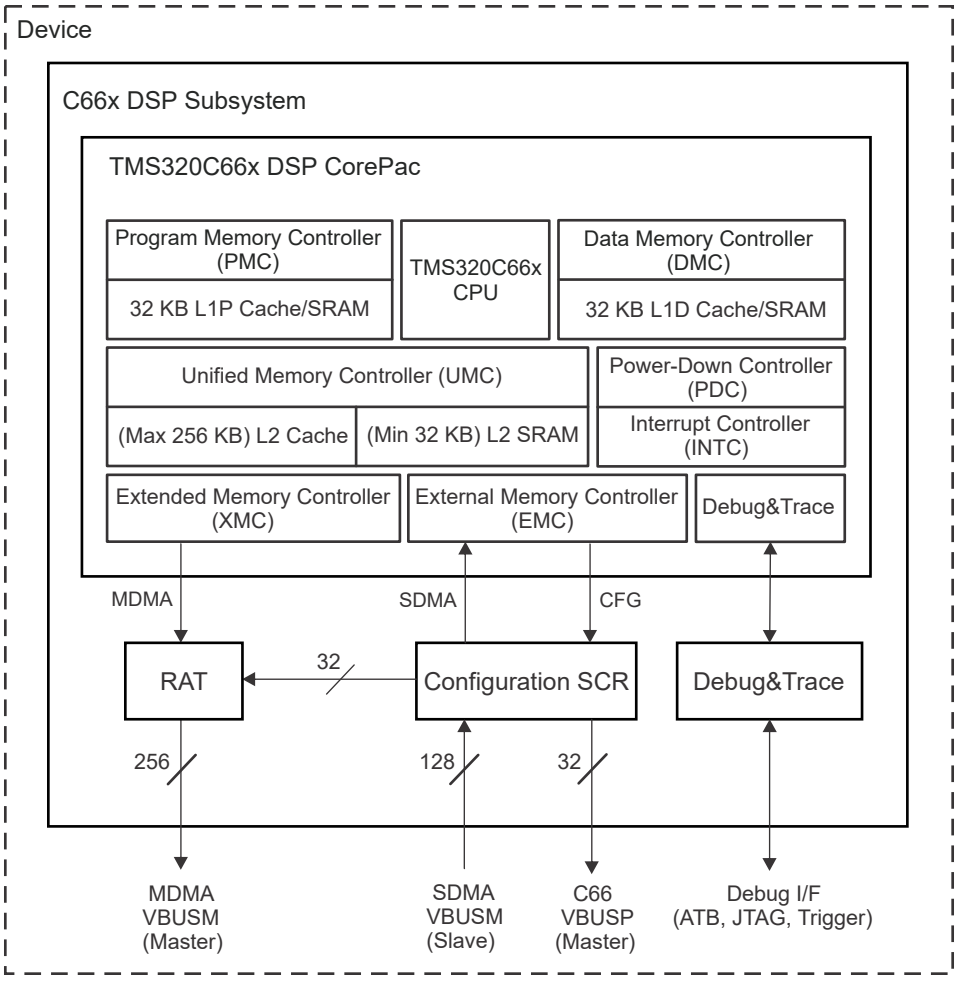

图 2-4. C66SS 概述

2.7.2 相关链接

- *TMS320C66x DSP CorePac 用户指南*
- *TMS320C66x DSP 高速缓存用户指南*
- 有关更多信息，请参阅 *DRA829/TDA4VM 技术参考手册* 中的 *C66x DSP 子系统* 一章。

2.8 TI 的 C7x 子系统

TMS320C71x 是下一代定点和浮点 DSP 平台。C71x DSP 是德州仪器 (TI) DSP 系列中的新内核。C71x DSP 支持矢量信号处理，与 C6x DSP 系列相比，能够在执行各种通用信号处理任务时显著提升 DSP 处理能力。此外，C71x 还提供多种专用函数，可将目标功能加快 30 倍以上。除了扩展矢量处理能力外，新型 C71x 内核还集成了高级技术，可提高控制代码效率并简化编程，例如分支预测、受保护的流水线、精确异常和虚拟存储器管理。

2.8.1 存储器布局

- 1 级 (L1)：
  - 具有 32KB L1P 存储器、所有高速缓存的 L1 程序存储器控制器 (PMC)（不支持 L1P SRAM）

  - 具有 48KB L1D 存储器的 L1 数据存储器控制器 (DMC)，可配置为高速缓存和/或 SRAM。例如，对于 TDA4VM，32KB 是高速缓存，剩余的 16KB 将是用于查找表实现的 SRAM。
- 2 级 (L2)：
  - 具有 512KB L2 存储器的 L2 统一存储器控制器 (UMC)，可配置为高速缓存和/或 SRAM
  - 在 SDK 中，默认情况下，它配置为 64KB 高速缓存和 448KB SRAM。
- L1D 高速缓存、SE、L2 SRAM、MSMC SRAM 和 DDR 之间完全一致
- MSMC 存储器可分配给 C71x 以提高性能。SDK 默认为为 C71x 分配最大可用 MSMC SRAM。

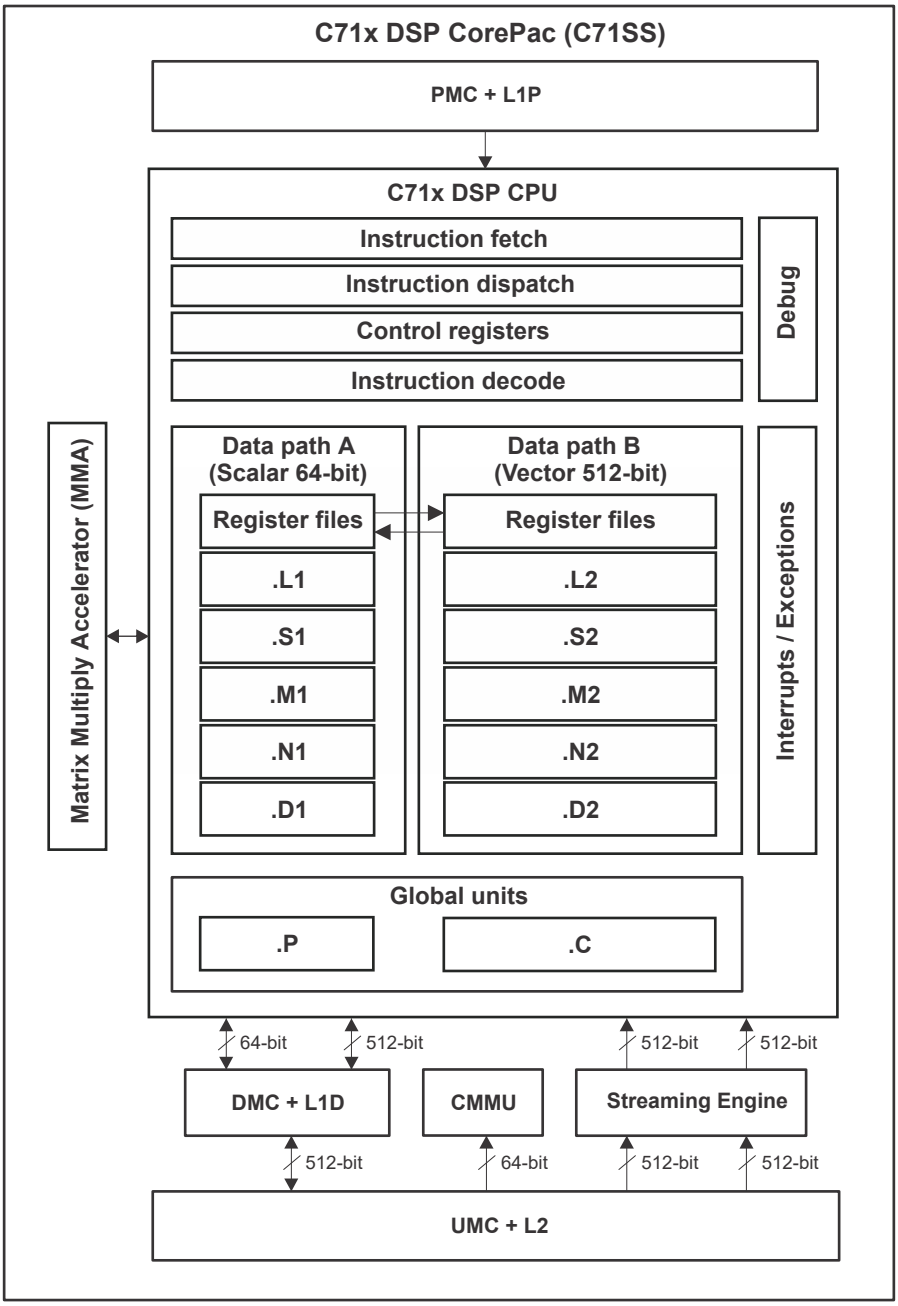

图 2-5. C71SS 概述

2.8.2 相关链接

- 有关更多信息，请参阅 *DRA829/TDA4VM 技术参考手册* 中的 *C71x DSP 子系统* 一章。
- *C7000 C/C++ 优化编译器用户指南*
- *C7000 优化指南*

- *C7000 主机仿真指南*
- *C6000 至 C7000 迁移用户指南*
- 学习软件流水线反馈信息 - 视频

2.9 DDR 子系统

此器件中的 DDR 子系统包含 DDR 控制器、DDR PHY 和包装器逻辑，用于将这些块集成到器件中。DDR 子系统被称为 DDRSS0，用于提供与外部 SDRAM 器件的接口，这些器件可用于存储程序或数据。DDRSS0 通过 MSMC 访问，而不是直接通过系统互连访问。

2.9.1 相关链接

- 有关更多信息，请参阅 *DRA829/TDA4VM 技术参考手册* 中的 *DDR 子系统 (DDRSS)* 一章。

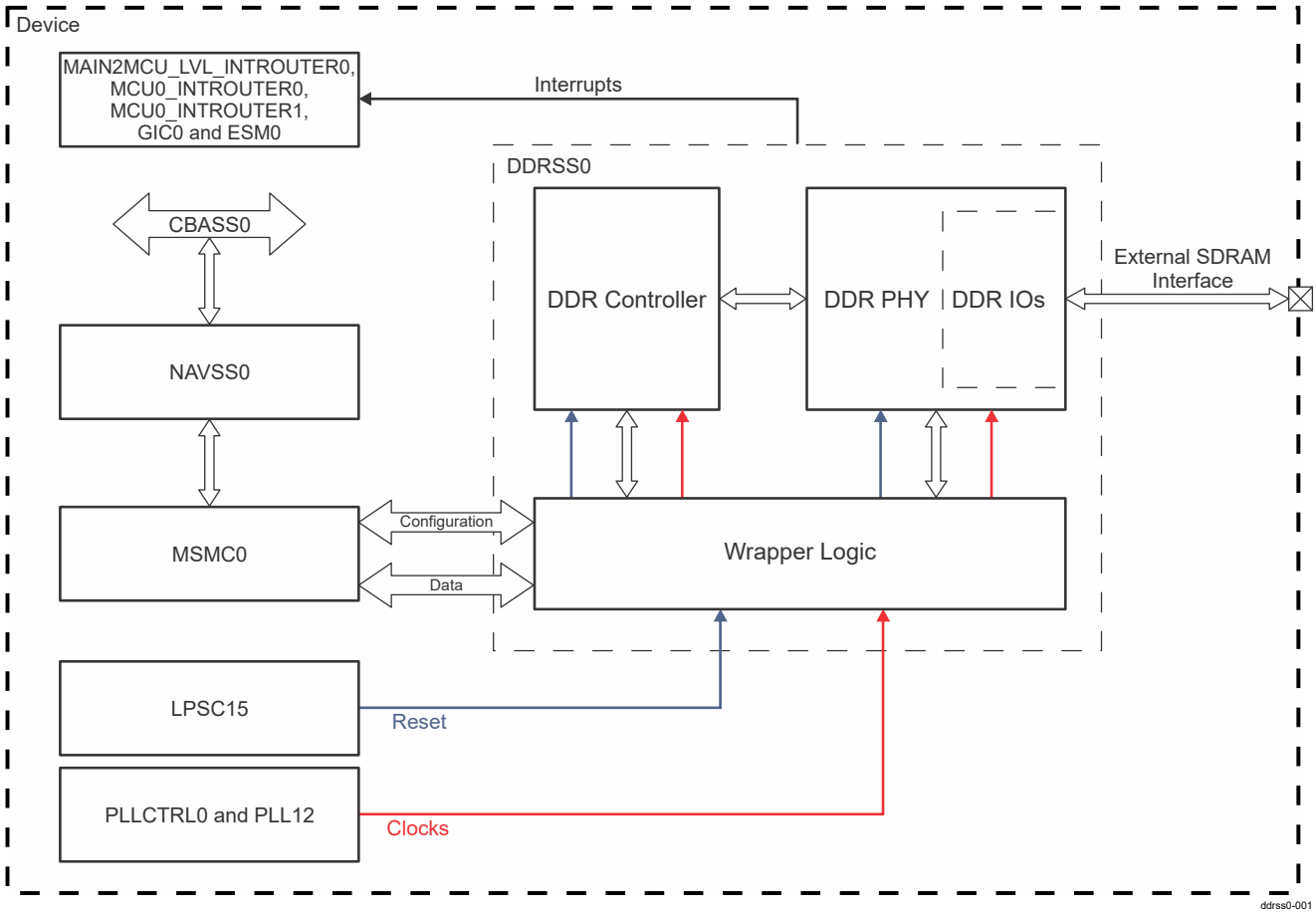

图 2-6. DDRSS 概述

3 性能数据

3.1 SDK 数据表

TI 的 SDK 版本在 SDK 文档中发布了一些基准测试。有关 SDK 性能的详细信息，请参阅以下资源。

- Linux SDK 性能指南：链接
- RTOS SDK 数据表：链接

请注意，这些链接可能会在发布时更改发布版本，但可以从 TI.com 上的 PROCESSOR-SDK-J721E 中导航至这些链接。

3.2 存储器访问延迟

表 3-1 和表 3-2 说明了 SoC 系统中每个内核到主存储器端点的相对存储器访问延迟（读取）。表 3-1 显示了内核访问不同存储器端点的延迟，以对 DDR 的访问作为基准。例如，在表 3-1 的第一行中，A72 内核访问 MSMC 的延迟是访问 DDR 的延迟的 33%。MCU OCRAM 和 MAIN OCRAM 的延迟都大于 DDR 的延迟。表 3-2 比较了从每个内核到 DDR 的延迟，以 A72 内核对 DDR 的访问作为基准。MCU R5 内核访问 DDR 所需的时间比 A72 内核长 2.55 倍。

应该注意的是，由于 SDRAM 刷新周期和定期再培训等因素，DDR 访问延迟通常不是恒定的。为了提供准确的比较，该分析通过排除非确定性因素来使用最佳 DDR 访问延迟。其他基于 SRAM 的存储器端点在访问延迟方面没有差异。

总的来说，这些表可让您深入了解系统中每个内核的相对存储器访问延迟，以及对系统性能和优化产生的影响。

请注意，表 3-2 中的相对性能适用于最坏情况下的延迟，例如，从物理内存而不是高速缓存读取数据时。如果正在处理的数据具有很大的局部特性，则高速缓存将隐藏大部分延迟。随着高速缓存未命中率的增加，使用不同存储器的性能提升开始显现。根据缓存和争用情况，从源到目标的观察到的性能会有所不同。SDK 数据表使用 LMBench（适用于在 A72 上运行的 Linux）和内存基准测试应用程序（适用于在 R5F 上运行的 FreeRTOS）等基准，比较了缓存的性能。

表 3-1. 从每个内核到存储器端点的相对延迟

	DDR	MSMC	C7x L2SRAM	C6x L2SRAM	MCU OCRAM	MAIN OCRAM
A72	1.00x	0.33x	0.38x	1.30x	1.59x	1.13x
C7x	1.00y	0.36y	0.03y	1.24y	1.50y	1.08y
C6x	1.00z	0.63z	0.67z	0.01z	0.52z	0.28z
MCU R5F	1.00a	0.73a	0.78a	0.54a	0.20a	0.47a
MAIN R5F	1.00b	0.71b	0.75b	0.51b	0.53b	0.42b

表 3-2. 每个内核的 DDR 访问延迟比较

	A72	C7x	C6x	MCU R5F	MAIN R5F
DDR	1.00c	1.12c	1.98c	2.55c	2.37c

从以上各表中可以看出，在系统设计过程中应考虑的内容：

- 与 DDR 相比，使用运行软件的处理器的本地存储器可降低存储器访问延迟
- 当使用存储器在内核之间共享数据时 (A72/R5/C6/C7)
  - 存储器访问时间将因软件运行的内核而异。
  - 在选择将使用哪个存储器来共享数据时应小心谨慎
- 与 DDR 相比，MSMC 存储器所有内核的存取时间都很短。在许多用例中，使用 MSMC 存储器均可提高性能。应审查在整个系统中高效使用该存储器的情况。

4 使用不同存储器时的软件注意事项

4.1 如何修改 RTOS 固件的存储器映射

MCU R5F、MAIN R5F、C66x、C7x 在默认 SDK 中运行 FreeRTOS。这些 FreeRTOS 固件的存储器映射可使用 SDK 中的链接器命令文件进行更改。请参阅此处的示例链接器命令文件以及此处和此处的头文件。请注意，更改存储器映射后，始终执行应用程序清理构建。

从链接器命令文件所做的更改会反映在生成的映射文件 (*.map) 中。映射文件可以位于构建环境的生成的二进制文件夹中。在 Linux 系统上，您可以在 elf 映像上运行“readelf -l”，以生成映射文件视图。

请注意，应使用所有系统共享存储器，使不同的内核不会相互破坏存储器空间。建议在电子表格或类似工具中创建系统存储器映射设计，并查看是否存在重叠。

4.2 RTOS 内核和 HLOS 之间的 DDR 共享

DDR 是系统中所有内核（R5F、M3、C6x、C7x、A72）之间的共享资源，需要进行分区以供不同内核使用。

DDR 使用情况：

- 不同内核固件的代码/数据
- 用于满足其动态存储器要求的 HLOS
- 内核之间的 IPC

要了解 vision_apps + Linux/QNX 用例的系统存储器映射，请参阅此开发人员手册。系统生成的存储器映射也可以在以下路径的自动生成文件中看到：*vision_apps/platform/j721e/rtos/system_memory_map.html*

用于 RTOS 固件的 DDR 存储器范围彼此不应重叠，并且还应该在 HLOS 侧保留。

- 在 Linux 系统中，可以使用器件树“reserved_memory”节点。请查看此处的示例。
- 如果使用 QNX，可以在此处查看有关对系统存储器映射进行更新的说明文件。

如果用于 MAIN R5F 的 DDR 区域（假设）未从 HLOS 中保留，则 HLOS 可能会使用该空间，从而导致存储器损坏，因为 MAIN R5F 软件在运行时会认为该 DDR 存储器范围仅供其使用。

4.3 引导加载程序使用的 MCU 片上 RAM

MCU_MSRAM 或 OCMC 由引导加载程序 SPL/SBL 使用，因此对应用程序使用的 MCU_MSRAM 或 OCMC 有一些限制。

有关更多信息，请参阅此处的开发人员手册以了解 OCMC 的用法。表 4-1 显示了这些详细信息。

表 4-1. 使用 MCU 片上 RAM 的存储器映射注意事项

地址	容量	用途	持久性
0x41C00000	512KB	保持 SPL/SBL 图像	仅在 SPL/SBL 执行期间。跳转到 MCU1_0 应用后即可声明。
0x41C80000	8KB	保持 SPL/SBL 和 SCISERVER APP 之间传递的电路板配置。	MCU R5F 应用在 MCU R5F 上执行 Sciclient_init() 后即可声明。
0x41C82000	502 KB	MCU 片上 RAM 上的可加载应用程序（包括 SCISERVER APP）	在整个 MCU R5F 应用程序执行过程中。
0x41CFFB00	约为 1.2KB	ROM 组合图像格式中使用的公共标头。	MCU R5F 应用在 MCU R5F 上执行 Sciclient_init() 后即可声明。

4.4 MSMC RAM 默认 SDK 使用情况

4.4.1 MSMC RAM 保留的段

对于 TDA4VM (J721E)，存在 8MB 的 MSMC RAM 并映射到 0x7000_0000。开头的 128KB 和结尾时 64KB 分别被 ATF 和 DMSC（用于 IPC）保留，因此不应使用。

表 4-2. MSMC RAM 保留的段

名称	起始地址	终止地址	大小	属性	说明
MSMC_MPU1	0x70000000	0x7001FFFF	128.00 KB	RWIX	MSMC 保留用于 MPU1 上的 ATF (Arm Trusted Firmware)
可用 MSMC（默认 SDK 将其分配给 MSMC_C7x_1）	0x70020000	0x707E7FFF	7.78 KB	RWIX	MSMC for C7x_1
MSMC_DMSC	0x707F0000	0x707FFFFF	64.00 KB	RWIX	MSMC reserved for DMSC IPC

4.4.2 MSMC RAM 配置为高速缓存和 SRAM

MSMC RAM 配置为高速缓存还是 SRAM 由传递给设备管理器的通用板配置决定。有关如何执行此操作的详细信息，请参阅 SDK 文档中的开发人员手册，了解如何配置 K3 MSMC 存储器以用作 SRAM 或 L3 高速缓存？

默认 SDK 将整个 MSMC 配置为 SRAM，并将其提供给 C7x DSP 以运行深度学习算法。此配置可根据客户用例进行更改。

4.5 从 MCU R5F 中使用 ATCM

在 MCU R5F 的特殊情况下，ROM 代码将内核退出复位状态。这与系统中其他内核的引导方式不同。其他主内核 MAIN R5F 或 DSP 由 MCU R5F 或 A72 上运行的软件引导加载程序引导。

上电时没有为 R5F 启用 ATCM（或 TCM-A），ROM 代码也未明确启用 ATCM（或 TCM-A）。ROM 代码后出现的 R5 SPL 不会改变 R5F 的状态，因此不会启用 ATCM。这意味着，除非已明确启用，否则不应从 MCU R5F 应用程序中使用 ATMC。因此，BTCM 用于存储 MCU R5F 固件的复位矢量。

不过，SBL 与 R5 SPL 不同，它会明确启用 ATCM；如果 SBL 用作引导加载程序（而非 R5 SPL 和 u-boot），则可以使用 ATCM。

4.6 使用 DDR 从 R5F 执行代码

PORz 后，R5F 的默认 MPU（存储器保护单元）未设置为具有 DDR 的执行权限，请参阅表 7-1 Arm Cortex R5F TRM。

要从 DDR 执行代码，需要在启动代码的 MPU init (__mpu_init()) 函数中重新配置 MPU，从而为映射到 0x8000_0000 的 DDR 提供可执行权限。完成此操作后，可以从 DDR 执行代码。

5 总结

Jacinto7 系列器件上运行的软件可以使用许多存储器。了解这些存储器的位置以及它们的使用方式对于系统设计非常重要。

虽然 DDR 存储器可随时用于所有内核，但并非始终是最高效的选择。在选择存储信息的位置时，必须权衡可用存储器的容量与该存储器的性能。这些软件架构决策对于特定内核而言可能是唯一的，但也可能会对整个系统产生影响。

建议在设计阶段审查整个系统的存储器使用情况。TI 处理器的 E2E 论坛是就此主题开展任何对话的好去处。

重要声明和免责声明

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、某特定用途方面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功能安全、信息安全、监管或其他要求。

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 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/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。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265