Application Note

ZHCAG38 – 12. 2025

AM62x SBL 和 SPL 启动模式使用说明

Joe Shen & Han Tao *Central FAE*

摘要

Sitara™ AM62x 处理器是 TI 公司推出的新一代通用处理器， 基于片上多核异构架构进行设计，集成不同的应用运算核心，将不同的任务分配给对应的核进行并行处理以最大限度的发挥处理器的性能优势。AM62x 系列处理器主要面向工业，自动化控制，工业/汽车 HMI 人机界面显示，PLC 可编程逻辑控制单元等应用方向。这些工业和汽车应用对于处理器的运算能力需求很高，同时还需要兼顾实时性处理和恶劣环境下稳定工作的要求。TI 为了支持这些应用场景，基于 AM62x 系列处理器发布了 No OS 裸机运行和实时操作系统 FreeRTOS 的 MCU-PLUS SDK，基于开源 OS 的 Linux SDK，Android SDK，商用 QNX SDK 等多个 SDK (Software Develop kit)开发包。在这些 SDK 中，TI 提供了配合芯片 ROM 固化程序引导整个处理器的 SBL （Second Boot Loader）或者 SPL (Secondary Program Loader) 两种引导软件。

本应用笔记将基于 AM62x 11.01 版本的 SDK 开发包，介绍关于 Sitara AM62x 系列处理器芯片，基于开发过程中大家常用的 SD card/eMMC 介质启动方法，详细讨论了配合 SBL 和 SPL 两种方式引导 AM62x 异构的片上 MCU 控制器和 Cortex-A53 HLOS （High Level Operation System）操作系统和应用程序的操作过程。并且总结了 SBL/SPL 开发过程中编译出来的各个运行应用程序的功能和生成文件所在路径，修订了一些 11.01 软件开发包使用指南中的命令路径错误，提供了在 SK-AM62-LP EVM 板上的 SBL 和 SPL 烧写测试方法供大家参考，希望通过这个应用笔记能够减少工程师在实际开发 AM62x 引导软件中的工作量。

目录

1. AM62x SBL 和 SPL 引导软件介绍

AM62x SOC 芯片提供了三种引导 SOC 的方法，第一种 no boot 模式需要通过外部的 JTAG 硬件接口，使用 TI 的硬件仿真器调试工具或者 ARM 的硬件仿真器来引导 AM62x 进行开发调试，这种软件引导模式在实际用户开发工作中很少使用。另外两种分别是基于开源 U-boot 项目适配 AM62x 的 SPL 和 TI 自己开发维护的 SBL 引导软件。引导软件 SPL 是在 Processor-SDK-LINUX-AM62x 发布包中提供的二级引导程序，它是基于开源项目 Uboot 的源程序，TI 适配 AM62x 芯片做了对应的修改以源程序方式提供。SPL 主要作用是初始化 AM62x 基本硬件并加载 UBOOT/Linux 主程序到内存中。SBL 软件是 MCU-PLUS SDK 中提供的二级引导程序。两种 bootloader 完成了 AM62x 的芯片和外设初始化，将 AM62x 内部各个运行内核配置为 WFI （Waiting For Interrupt）可运行状态。针对不同的引导介质在 SDK 中会编译出不同的 SPL/SBL 引导 image。所以实际项目开发过程中不同用户会基于 SDK 维护各自不同的 SPL/SBL 工程文件。

由于 AM62x 会适配不同容量，不同厂家不同型号的 DDR4/LPDDR4, 所以第一步 SPL/SBL 将被 ROM 代码装载到 AM62x 片上的 SRAM 空间运行，通过 SBL/SPL 初始化外部的 LPDDR4/DDR4 颗粒后，才能够继续将操作系统等文件大小超过片上 SRAM 的容量，image size 更大的可执行程序例如 U-boot，kernel 或者其他的 R5F 核的程序 image 装载到外部的 DDR4/LPDDR4 继续运行。

1.1 AM62x 使用的 SBL 和 SPL 两种引导程序的对比

在实际开发工作中很多用户会碰到两种引导模式的选择，下面就简单对比了一下 SBL 引导和 SPL 引导的主要区别，以方便大家在项目开发过程中选择适合自己的启动方式。

	SBL	SPL
开发维护	TI 开发和维护	TI 基于开源社区 Uboot 发布版本适配 AM62x
适配芯片	支持 TI 的 MCU，Sitara MPU, Jacinto SOC	支持 TI 的 Sitara MPU, Jacinto SOC
是否支持加密启动	支持	支持
SDK 适配 OS	AutoSAR, QNX, Vxworks, Linux,FreeRTOS	Linux,Android
异构内核加载	支持加载所有异构核 image	加载 A53 核运行的 uboot 或者 Linux kernel
支持的启动设备	UART, SD card, eMMC, SPI, OSPI, GPMC	UART, SD card, eMMC, SPI, OSPI, USB , Ethernet
发布所在软件包	MCU-PLUS-SDK 软件包	PROCESSOR-SDK-LINUX-AM62X 软件包

表 1 SBL 和 SPL 对比表

总结：如果在 A53 核上运行 Linux 或者 Android 操作系统，使用 SPL 作为 bootloader 会更加常用。作为原生开源 Uboot 项目的一部分，能够复用很多 Uboot 开发的经验。如果需要上电后同时启动 AM62x 内部的 Wakeup R5F 核, MCU R5F 核 和 A53 核，需要减少 Linux 启动时间，或者需要在 A53 核上运行 QNX，FreeRTOS，Vxworks 等其他的操作系统，那么基于 SBL 启动会更适合。

1.2 AM62x 启动设备的分区说明

AM62x 支持多种启动模式，在 AM62x 的 TRM （Technical Reference Manual）里面有 11 种启动方法的详细介绍。不同的启动方式分别适用于开发调试，工厂生产，量产使用，售后维修等不同的使用场景。下面根据用户在项

目开发阶段经常使用的 SD card/eMMC 两种介质启动方法，介绍了配合 SBL/SPL 引导时候的使用注意事项和开发过程中需要注意的细节。

由于 SBL/SPL 都是通过 AM62x 芯片内部固化的 ROM（Read Only Memory）来加载，因此对于 eMMC/SD card 引导时候的文件放置位置和文件名称都有固定的要求。这个文档是按照开发过程中常用的 SD card/eMMC 启动模式，介绍了在 SDK11.01 中默认的分区方法和引导文件的文件名称，放置位置和使用方法做的说明。这部分是和 SBL/SPL 启动软件开发包中代码默认设置相匹配，所以需要一一对应。TI 的 SDK 开发包中 SBL 和 SPL 都是以源代码模式提供，用户可以根据实际使用中 image 大小的变化来进行修改。

右图是使用 balenaEtcher 烧录工具基于 tisdk-default-image-am62xx-evm-11.01.05.03.rootfs.wic.xz 开发软件包制作的 SD 卡分区，ROM 代码通过 FAT 文件系统读取 SD 卡上的 SBL 或者 SPL 文件完成 image 的加载和 AM62x 处理器的引导。SBL 通过 SD 卡启动时可以使用和 SPL 一样的 SD 卡分区，除了用 balenaEtcher 工具来制作可以启动 AM62x 的 SD 卡，可以按照参考文献 4 中第一章节的方法来手工制作启动 SD 卡。

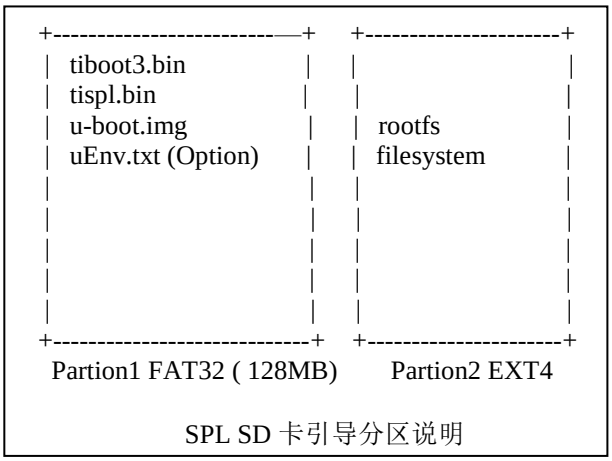

图 1 SD 卡分区说明

2. SBL 和 SPL 在 SDK 包中生成文件的位置和烧写命令

在原生 SDK 开发过程中，TI 建议用 Ubuntu 22.04 作为开发环境。只有 SBL 支持 windows 开发环境，SPL 只支持 Linux 开发环境开发，建议用户使用和 TI 原生 SDK 相同的 Linux 开发环境，以避免由于 Linux 版本不同导致的一些未验证过的编译环境和库依赖问题。因此后面的文件位置和烧写命令都是基于 Ubuntu 22.04 发布版本的 Linux 开发环境所做的测试，所以如果是 windows 开发环境请参考 MCU-PLUS-SDK 的开发说明。

2.1SPL 生成结果在 SDK 包中的默认位置和烧写方法

在 SK-AM62x-LP EVM 板上开发验证的时候，默认 SDK 的安装路径是 TI_SDK_PATH?=~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03 ，这个路径可以在 SDK11.1 的安装目录下的 Rules.make 中修改成项目开发中工程安装的特殊路径，后面为了简洁都是基于默认的 TI_SDK_PATH 目录为~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03 的情况下进行说明。

Image 的功能	SPL 生成路径
ROM 加载 R5F 第一个 image	./board-support/ti-u-boot-2025.01+git/out/r5/tiboot3.bin
A53 核加载的第一个 image	./board-support/ti-u-boot-2025.01+git/out/a53/tispl.bin
A53 核加载 U-boot 程序	./board-support/u-boot-build/a53/u-boot.img

表 2 SPL 编译生成可执行文件路径表

在 Linux SDK 的安装目录运行 make u-boot 命令会生成 SPL 启动的 image。对应几个启动 image 的加载顺序说明在参考文献 4 的第 3.1.1.4 章节有详细介绍。这三个文件可以直接通过拷贝到 SD 卡的 boot 分区对应文件的方式来更新。更简单的烧写方法是用 SDK 里面的 makefile 来直接更新这三个文件。命令中的 DESTDIR 宏是 SD 卡挂载到 ubuntu 上对应的目录。

```
#make u-boot
#sudo DESTDIR=/media/tao/boot make u-boot_install
```

2.2 SBL 生成的文件和烧写方法

SBL 默认安装所在路径为 MCU_PLUS_SDK_PATH ?= ~/mcu_plus_sdk_am62x_11_01_00_16。MCU-PLUS SDK 的开发目录路径在安装目录的 makefile 第一行进行声明，在实际开发工作中可以根据项目的代码管理需求对应修改 SDK 实际安装路径。由于 SBL 可以作为裸 SOC 运行的 NoRTOS 程序，RTOS 操作系统，AutoSAR 操作系统和 linux 操作系统的 bootloader，为了简要说明 SBL 的使用方法本应用笔记用 SBL 引导 Linux 操作系统作为参考来介绍。通过 SBL 来启动其他 OS 的方法可以按照参考文档 3 的说明来操作。由于 SBL 生成的文件有 7 个，对应的文件所在位置详细信息在 MCU_PLUS_SDK 的安装目录下的配置文件/tools/boot/sbl_prebuilt/am62x-sk-lp/default_sbl_emmc_hs_fs.cfg 中有详细路径和文件名称。

Image 的功能	SBL 生成文件名
ROM 加载 R5F 第一个文件，等同于 SPL 的 tiboot3.bin	sbl_emmc_linux_stage1.release.hs_fs.tiimage
R5F 上第二个运行文件，用于运行 HSM M4，SPL 程序	sbl_emmc_linux_stage2.release.appimage.hs_fs
MCU M4 核上运行的 IPC 示例	/m4fss0-0_freertos/ti-arm-clang/ipc_rpmsg_echo_linux.release.appimage.hs_fs
HSM M4 核上运行的 HSM 模块，该 M4 和 MCU M4 是不同的核	hsm.appimage.hs
DM R5F 核上运功的 IPC 示例	/r5fss0-0_freertos/ti-arm-clang/ ipc_rpmsg_echo_linux.release.appimage.hs_fs
A53 核运行的 u-boot	u-boot.img
Linux kernel	linux.appimage.hs_fs

表 3 SBL 编译生成运行 Linux 操作系统的文件路径表

由于 SBL 启动 Linux 涉及到 SBL 的 bootloader 的编译和程序生成，又涉及到 Linux 操作系统的文件通过工具产生 SBL 可以直接加载的可执行二进制文件，所以 eMMC 引导模式 SBL 启动 Linux 的 image 生成会繁琐一些。下面的编译命令需要工程师特别小心编译 makefile 的配置，如果配置不对的话编译和烧写不能正常运行。下面是详细步骤介绍 SBL 启动 Linux 所需要做的准备工作：

第一步：首先需要编译 MCU 整个 SDK，里面会包含 SBL 启动的可执行程序和 AM62x 异构核上的示例程序。
第二步：单独生成 HSM M4 上运行的 HSM Appimage
第三步：使用脚本工具将 Linux 开发环境中的生成的 BL31.bin，BL32.bin，tispl.bin 和 kernel 的 Image 文件打包生成 linux.appimage 执行文件。
第四步：将 EVM 板更改为 UART 引导模式，通过 UART 端口烧写 image 到 eMMC。
第五步：重新配置 EVM 板引导模式为 eMMC，SBL 启动 U-boot 和 Linux 进行测试。

默认 SDK 测试常见问题

问题 1：按照 Linux SDK11.01 使用指南直接编译的 SPL 停留在启动位置。如下面出错信息截图

```
// 在 MCU-PLUS-SDK 安装目录下编译生成 SBL 引导程序
#make -s all

//在 MCU-PLUS-SDK 安装目录下修改 linux 开发环境的运行目录
//例如 MCU-PLUS-SDK 安装到了目录~/mcu_plus_sdk_am62x_11_01_00_16
// Linux 开发环境安装到了目录~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03
//修改~/mcu_plus_sdk_am62x_11_01_00_16/tools/boot/linuxAppimageGen/board/am62x-sk-lp/config.mak 配置文件
//将 config.mak 配置文件第六行 PSDK_LINUX_PATH=$ 和第九行的 PSDK_LINUX_PREBUILT_IMAGES?=
//宏定义改为
//PSDK_LINUX_PATH=~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03
//PSDK_LINUX_PREBUILT_IMAGES?= =~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03/board-
support/prebuilt-images/am62xx-lp-evm

#cd  ~/mcu_plus_sdk_am62x_11_01_00_16/tools/boot/linuxAppimageGen
# make -s BOARD=am62x-sk-lp all

//通过 uart boot 模式启动 AM62x-LP-EVM 板，并将生成 SBL+Linux boot image 烧写到 emmc 里面。
#cd ~/mcu_plus_sdk_am62x_11_01_00_16/tools/boot
# python uart_uniflash.py -p /dev/ttyUSB1 --cfg=./sbl_prebuilt/am62x-sk-lp/default_sbl_emmc_linux_hs_fs.cfg
```

【回答】默认的 SDK 是适配 AM62x-SK EVM 板，目前常用的评估板是 AM62x-SK EVM 板。需要用下面的编译命令重新编译 U-boot

```
# make u-boot PLATFORM=am62xx-lp-evm
```

```
U-Boot SPL 2025.01-gd2a72467939e (Aug 07 2025 - 15:00:56 +0800)
SYSFW ABI: 4.0 (firmware rev 0x000b '11.1.2--v11.01.02 (Fancy Rat)')
Changed A53 CPU frequency to 1250000000Hz (T grade) in DT
SPL initial stack usage: 13392 bytes
```

问题 2：按照脚本烧写 SBL 启动文件时会报找不到 hsm.appimage.hs_fs 文件的错误

【回答】默认的 MCU-PLUS-SDK 开发包中不包含 HSM 示例运行程序，需要在 MCU-PLUS_SDK 的安装目录中进入到 HSMAppimageGen 目录来编译生成可执行程序

```
PATH:       /mcu_plus_sdk_am62x_11_01_00_16/tools/boot/HSMAppimageGen/
Command:
            # cd  /tools/boot/HSMAppimageGen/
            # make BOARD=am62x-sk-lp all
```

问题 3：如何生成 SBL 引导模式需要的 BL31.bin 文件。

【回答】按照 SDK11.1 的 user guide 编译 BL31.bin 文件会出现工具链找不到的问题，需要配置 Linux 的环境变量以使用正确的工具链链编译 BL31.bin。下面是配置环境变量的脚本，方便大家生成自己的 BL31.bin 文件。

Step1: 在 Linux 开发环境运行下面的脚本或者将所有的命令拷贝出来单独运行。

Step2: 进入 ARM Trusted Firmware-A 的安装路径，并编译出 BL31.bin 文件。

Step3: 将编译生成的 bl31.bin 文件拷贝到 SDK11.1 默认 u-boot 编译时使用的位置

下面是编译 BL31.bin 的环境变量配置。

默认 SBL 引导 Linux 固件生成工具默认的 BL31.bin 和 BL32.bin 文件读取路径：
/ti-processor-sdk-linux-am62xx-evm-11.01.05.03/board-support/prebuilt-images/am62xx-lp-evm/
重新编译 BL31.bin 文件的环境变量配置脚本：

```
#!/bin/bash
#setup AM62 11.01 uboot compile environment
export SDK_INSTALL_DIR="~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03"
export CROSS_COMPILE_64="${SDK_INSTALL_DIR}/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/aarch64-oe-linux/aarch64-oe-linux-"
export SYSROOT_64="${SDK_INSTALL_DIR}/linux-devkit/sysroots/aarch64-oe-linux"
export CC_64="${CROSS_COMPILE_64}gcc --sysroot=${SYSROOT_64}"
export CROSS_COMPILE_32="${SDK_INSTALL_DIR}/k3r5-devkit/sysroots/x86_64-arago-linux/usr/bin/arm-oe-eabi/arm-oe-eabi-"
export UBOOT_DIR="${SDK_INSTALL_DIR}/board-support/ti-u-boot-2025.01+git"
export TI_LINUX_FW_DIR="${SDK_INSTALL_DIR}/board-support/ti-linux-firmware"
export TFA_DIR="${SDK_INSTALL_DIR}/board-support/trusted-firmware-a-2.13+git"
export OPTEE_DIR="${SDK_INSTALL_DIR}/board-support/optee-os-4.6.0+git"
```

这是 BL31.bin 的编译命令

```
#cd ~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03/board-support/trusted-firmware-a-2.13+git/
#make ARCH=aarch64 CROSS_COMPILE="$CROSS_COMPILE_64" PLAT=k3 TARGET_BOARD=lite SPD=opted
#cp ./build/k3/lite/release/bl31.bin ~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03/board-support//am62xx-lp-evm/bl31.bin
```

问题 4：在编译 SBL 引导模式需要的 BL32.bin 文件时出现下面的错误如何解决。

```
LD      out/arm-plat-k3/core/all_objs.o
/home/tao/ti-processor-sdk-linux-am62xx-evm-11.01.05.03/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/aarch64-oe-linux/aarch64-oe-linux-ld.bfd: cannot find libgcc.a: No such file or directory
make: *** [core/arch/arm/kernel/link.mk:69: out/arm-plat-k3/core/all_objs.o] Error 1
```

【回答】按照 SDK11.1 的 user guide 时候编译 BL32.bin 文件会出现工具链中不包含默认的 libgcc.a 库文件。所以在编译 OPTEE 的 BL32.bin 文件就不能使用 SDK11.1 中自带的交叉编译工具链，需要用 ARM 官方发布的编译工具链来编译。为了方便大家使用，下面是修改 OPTEE 编译工具链的脚本。

```
#!/bin/bash
#setup AM62 11.01 uboot compile environment
 export CROSS_COMPILE_64=~/ti/arm-gnu-toolchain-13.3.rel1-x86_64-aarch64-none-linux-gnu/bin/aarch64-none-linux-gnu-
 export CROSS_COMPILE_32=~/ti/arm-gnu-toolchain-13.3.rel1-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-
 export CC_64="${CROSS_COMPILE_64}gcc"
 export CC_32="${CROSS_COMPILE_32}gcc"
```

Step1: 在 Linux 开发环境运行下面的脚本或者将所有的 export 配置环境变量命令拷贝出来运行。
Step2: 进入 OPTEE 的安装路径，并编译出 BL32.bin 文件。
Step3: 将编译生成的 bl32.bin 文件拷贝到 SDK11.1 默认 u-boot 编译时使用的位置。

```
#cd ~/ti-processor-sdk-linux-am62xx-evm-11.01.05.03/board-support/ optee-os-4.6.0+git/
#make CROSS_COMPILE="$CROSS_COMPILE_32" CROSS_COMPILE64="$CROSS_COMPILE_64" PLATFORM=k3-am62x CFG_ARM64_core=y
#cp ./out/arm-plat-k3/core/tee-pager_v2.bin ../prebuilt-images/am62xx-lp-evm/bl32.bin
```

问题 5：SBL/SPL 如何适配自己用的 DDR4/LPDDR4 颗粒。

【回答】TI 提供了工具帮助工程师们配置 LPDDR4/DDR4 的参数，在线配置工具链接是（https://dev.ti.com/sysconfig）。sysconfig 工具也提供离线下载软件包。配置工具方便用户在使用不同厂家的 DDR 颗粒时通过输入颗粒参数生成匹配最优参数结果，将参数结果编译到 SBL/SPL 中就能够适配不同的内存颗粒。工具中的参数可以在内存颗粒手册中逐项查找到，填写完成后工具会生成 SBL 使用的 board_lpddrReginit.h 文件和 spl 使用的参数配置文件。下面列出来了实际使用中针对不同类型的颗粒和不同的启动软件需要修改的参数文件位置。只要将 sysconfig 工具生成的文件替换下面 SBL/SPL 的对应文件重新编译后就能够适配新的内存颗粒。

```
SBL DDR4/LPDDR4 参数路径：
LPDDR4 参数文件路径    ./source/drivers/ddr/v0/soc/am62x/board_lpddrReginit.h
DDR4 参数文件路径       ./source/drivers/ddr/v0/soc/am62x/board_ddrReginit.h
SPL DDR4/LPDDR4 参数路径
LPDDR4 参数文件路径 . /arch/arm/dts/k3-am62-lp4-50-800-800.dtsi
DDR4 参数文件路径      ./arch/arm/dts/k3-am62x-sk-ddr4-1600MTs.dtsi
```

3. 总结

SBL 和 SPL 是 AM62x 处理器启动时第一步运行的外部程序，也是各位工程师开始使用 AM62x 芯片开发自己项目，运行自己设计的硬件电路板第一个需要运行的程序，希望这篇 SBL 和 SPL 使用文档能够帮助各位工程师开发过程中少走弯路。在实际开发过程中碰到的问题，也欢迎各位到 TI 技术论坛 e2e.ti.com 来一起讨论。

4. 参考文献

1. AM62x Sitara™ Processors datasheet (Rev. B)
2. AM62x Sitara Processors Technical Reference Manual (Rev. B)
3. MCU-PLUS-SDK-AM62X 11.01.00.16 online documentation
4. PROCESSOR-SDK-LINUX-AM62X 11.01.05.03 Yocto SDK user manual for developers
5. TI Processor engineer support forum

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