Application Note

使用 SPI 将 MSPM0 – ADC 连接到 AM62x 和 AM62L 上

Divyansh Mittal, Anshu Madwesh, Yashraj Motwani, Anil Swargam, Anshu Choudhary, Krunal Bhargav, Alisa, Thomas

摘要

本应用手册介绍了如何借助串行外设接口 (SPI) 将 MSMP0 上的 ADC 集成到 2 个充当控制器的处理器（AM62x 和 AM62L）中，以支持高速 ADC 数据传输。AM62x 是一款异构处理器，配备多达四个 Arm Cortex A53 处理器和一个 Arm Cortex M4F 内核。AM62L 是 AM62x 的精简版，具有两个 Arm Cortex A53 内核，不含 Arm Cortex M4 MCU 内核。AM62x 不带有板载 ADC，而 AM62L 具有大约 10 位 ENOB ADC。本文旨在演示将 MSMP0 微控制器的 ADC 集成到 AM62x 和 AM62L 中的过程。此操作可以在 AM62x 上启用 ADC 并为 AM62L 提供更高分辨率的选项。MSPM0 微控制器配备了一个多通道 ADC，通过该 ADC，我们可以监控多个模拟信号和传输任意/所有数字信号，以及通过 SPI 传输到 SoC。本文将进一步深入探讨总体数据流、硬件和软件设置、执行应用程序代码的步骤以及预期结果。

内容

商标

1 简介

1.1 SPI 事务数据流

我们配置 MSPM0L130x 微控制器上的 ADC，并通过 SPI 接口连接 AM62x SK/AM62L 微处理器。这里，处理器 - AM62x/AM62L 已配置为控制器，MSPM0L130x 已配置为外设。要获取 ADC 任一通道的数据，控制器可以使用 TX 缓冲区中的相应命令启动 SPI 事务。一接收到控制器命令，外设便开始在所请求的通道上传输加载到其 TX 缓冲区中的 ADC 数据。控制器从外设接收预期字节数，然后结束事务。外设持续读取和更新 ADC 数据值。这些更新的频率取决于用于触发 ADC 的计时器。 [1]

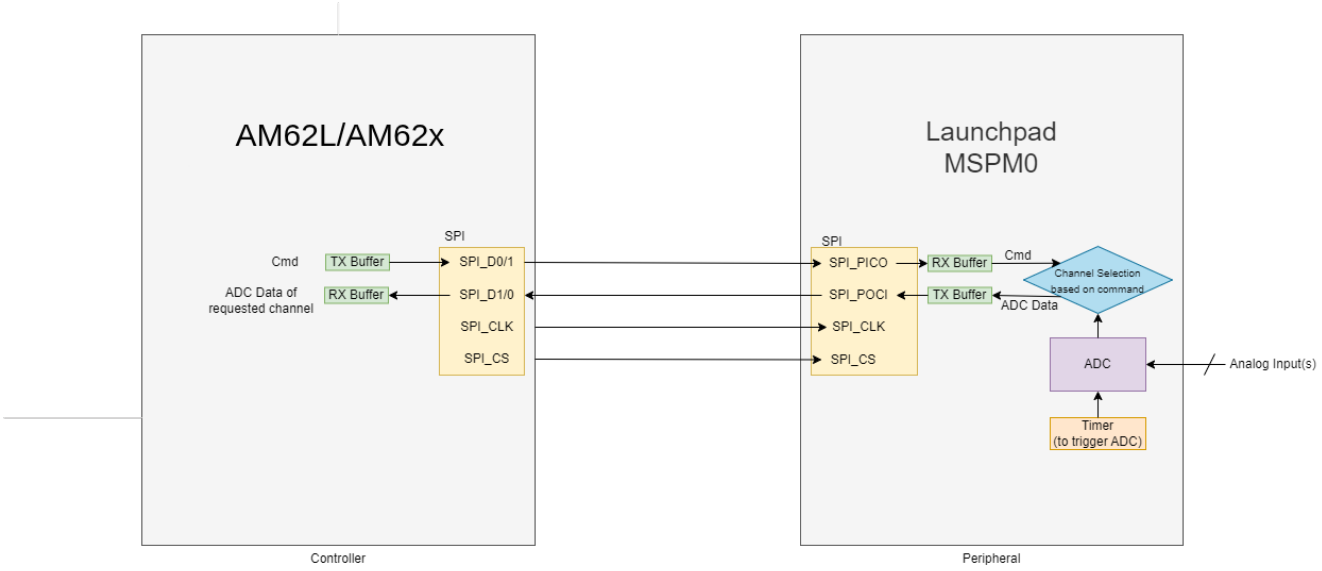


图 1-1. 控制器（SK-AM62x 或 AM62L EVM）和外设 (LPMSPM0L130x) 之间的总体数据流

使用多通道模式时全双工 SPI 的流水线：

在全双工 SPI 模式下，数据在同一组时钟周期内同时发送和接收。因此，在使用多通道 ADC 的情况下，当控制器发送命令时，它会同时接收与其上一条命令相对应的 ADC 数据。

运行此应用涉及的步骤如下：

1. 硬件设置包括控制器 - SK-AM62x/AM62L EVM 和外设 LP-MSPM0L130x 之间的连接。
2. 软件设置，包括一次性执行前步骤。
3. 在两个电路板上执行应用程序以启用 SPI 事务。
4. 结果分析。
5. 系统性能分析和功耗估算。

1.2 AM62x 和 AM62L 处理器

AM62x 处理器

AM62x Sitara 微处理器（如图 1-2 所示）是一种为各种嵌入式应用而设计的异构处理器。可通过 A53 内核上的 MAIN 域启用 SPI。图 1-2 显示了 AM62x 的简化方框图。

[1] 注意：这里不使用“主”和“从”以及“MOSI/MISO”术语，这些术语将分别替换为“控制器”和“外设”以及“PICO/POCI”。

如需了解更多详细信息，请参阅 AM62x Sitara 处理器数据表。

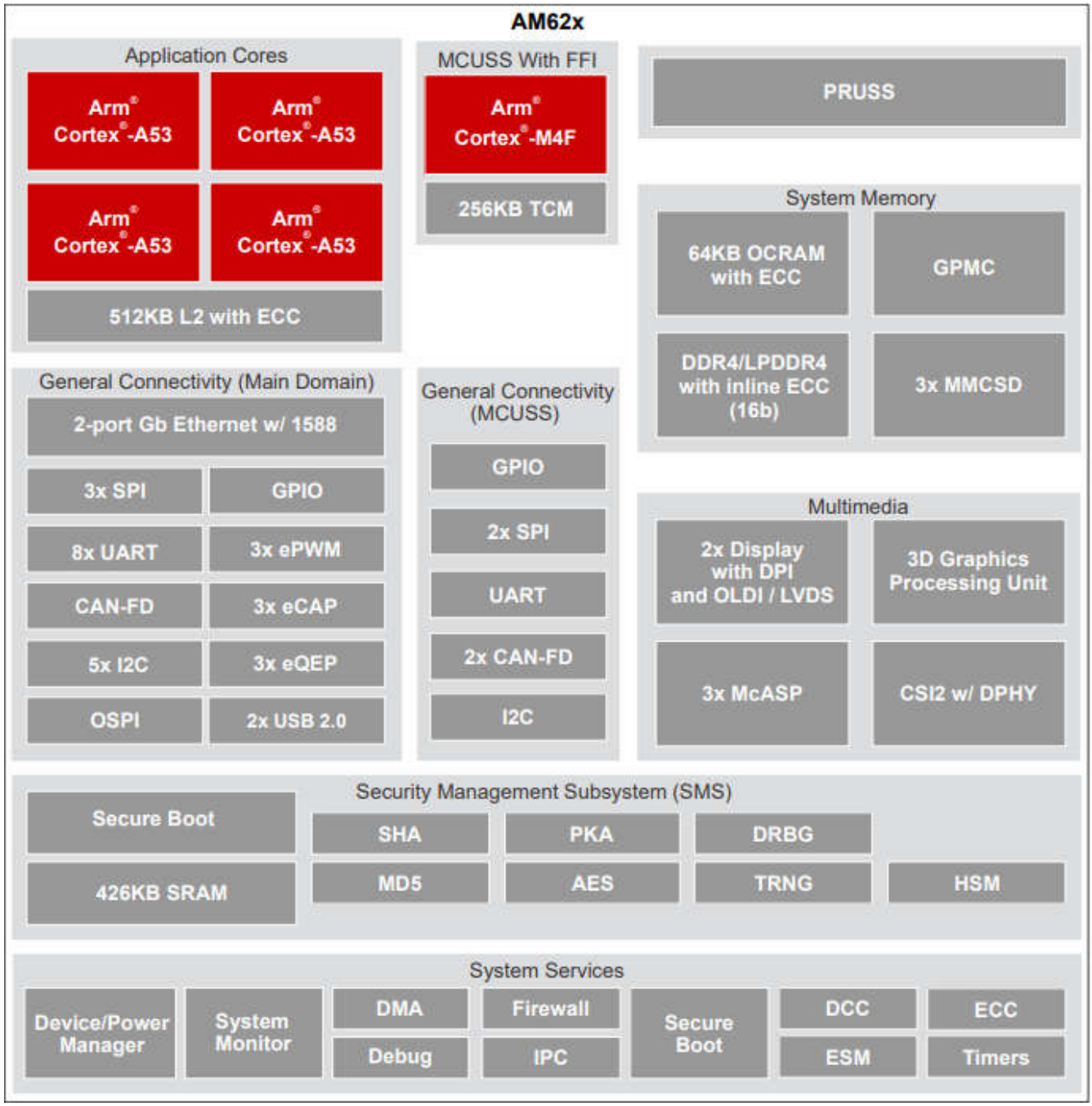


图 1-2. AM62x 简化方框图

AM62L 处理器

AM62L Sitara 微处理器（如图 1-3 所示）是 AM6x 系列中的一款低成本且性能经过优化的处理器。可通过 A53 内核上的 MAIN 域启用 SPI。图 1-3 显示了 AM62L 的简化方框图。

如需了解更多详细信息，请参阅 AM62L Sitara 处理器数据表。

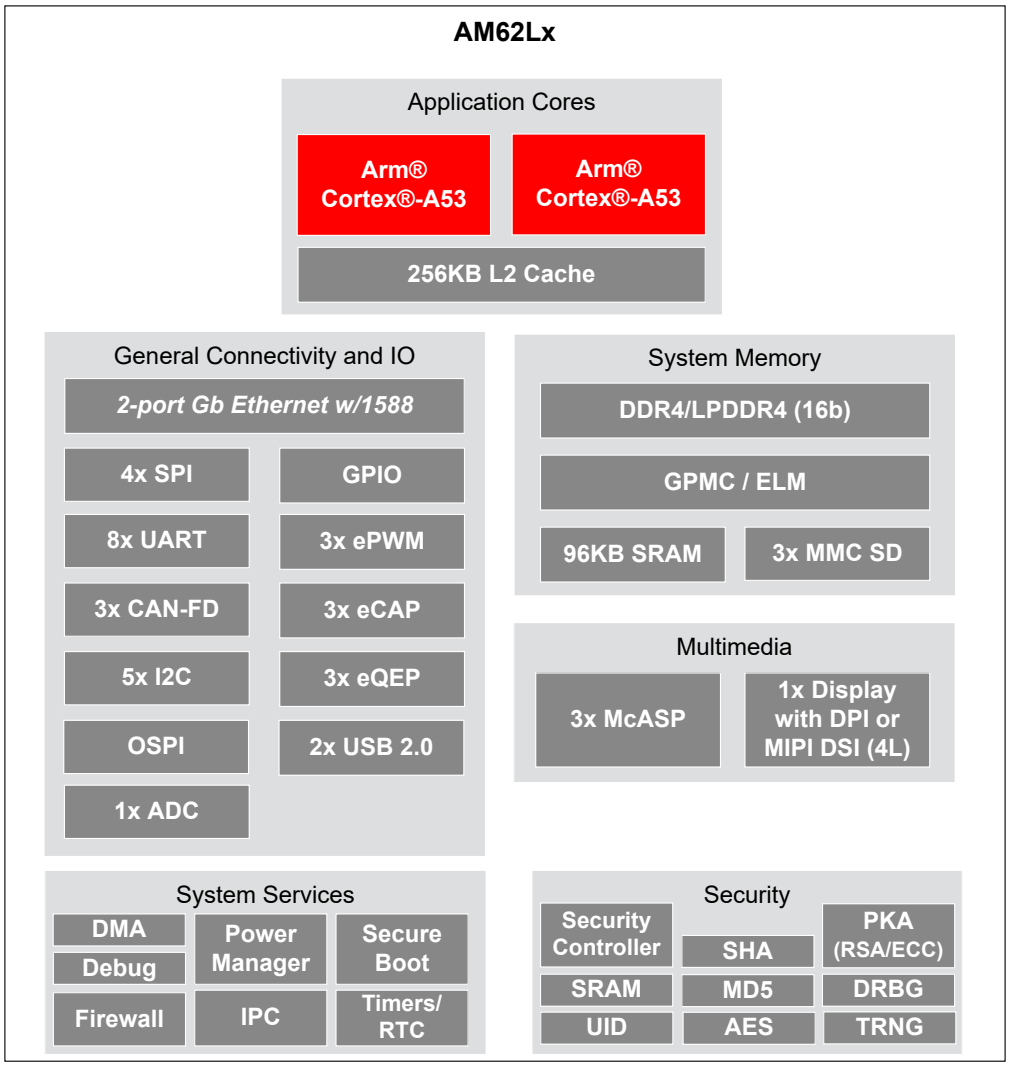


图 1-3. AM62L 简化方框图

1.3 MSPM0L130x 微控制器

MSPM0L130x 微控制器（如图 1-4 所示）是一款易于使用的评估模块 (EVM)。

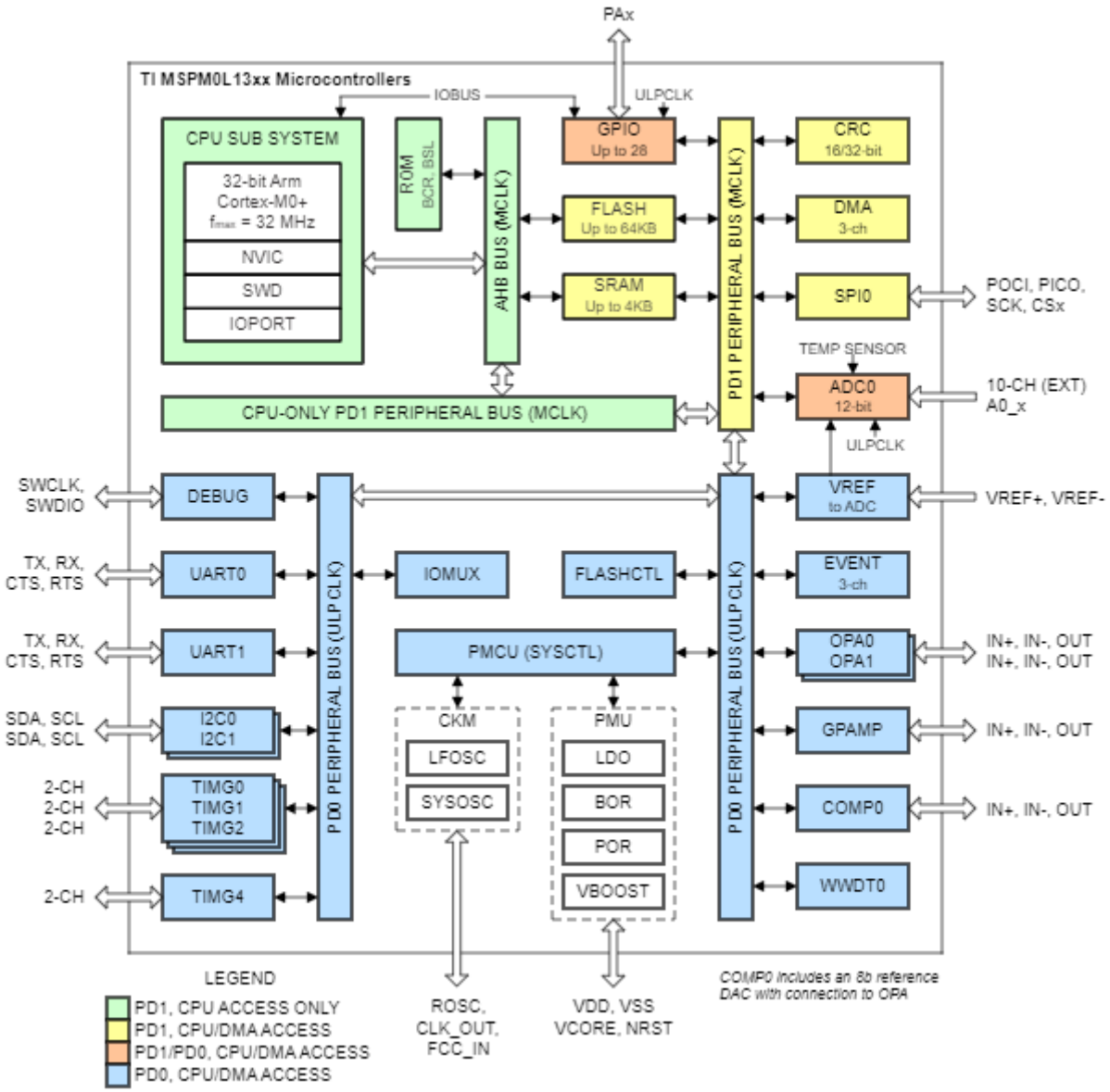


图 1-4. MSPM0L130x 简化方框图

M0L 器件上的主计算和接口子系统如下所示：

- Arm Cortex-M0+ 内核：此平台可以在高达 32MHz 的频率下运行。它是成本优化型 MCU，可提供高性能模拟外设集成。
- 板载 ADC 支持快速的 12 位、10 位和 8 位模数转换，具有 12 位 SAR 内核、采样和转换模式控制功能和多达 4 个独立的转换和控制缓冲器，并以 12 位分辨率提供 1.68Msps 转换速率
- 具有 SPI 模块，可在高达 16Mb/s 的速度下运行。

有关更多详细信息，请参阅 MSPM0L130x 微控制器数据表。

2 硬件设置

要运行应用程序代码，必须执行以下电缆连接。请注意，为这些连接选择的引脚适用于特定的 SPI 通道。如果对 SPI 通道或引脚多路复用进行了任何修改，则需要通过数据表检查相应的引脚，然后使用。

2.1 AM62x

AM62x 由 2 个域（4 个 A53 内核的 MAIN 域和 1 个 M4F 内核的 MCU 域）组成。显示了两个内核的硬件设置。

2.1.1 A53 内核硬件设置

为了使用 A53 内核，SK-AM62x 上的 SPI 外设引脚位于用户扩展接头中。图 2-1 显示了硬件设置。

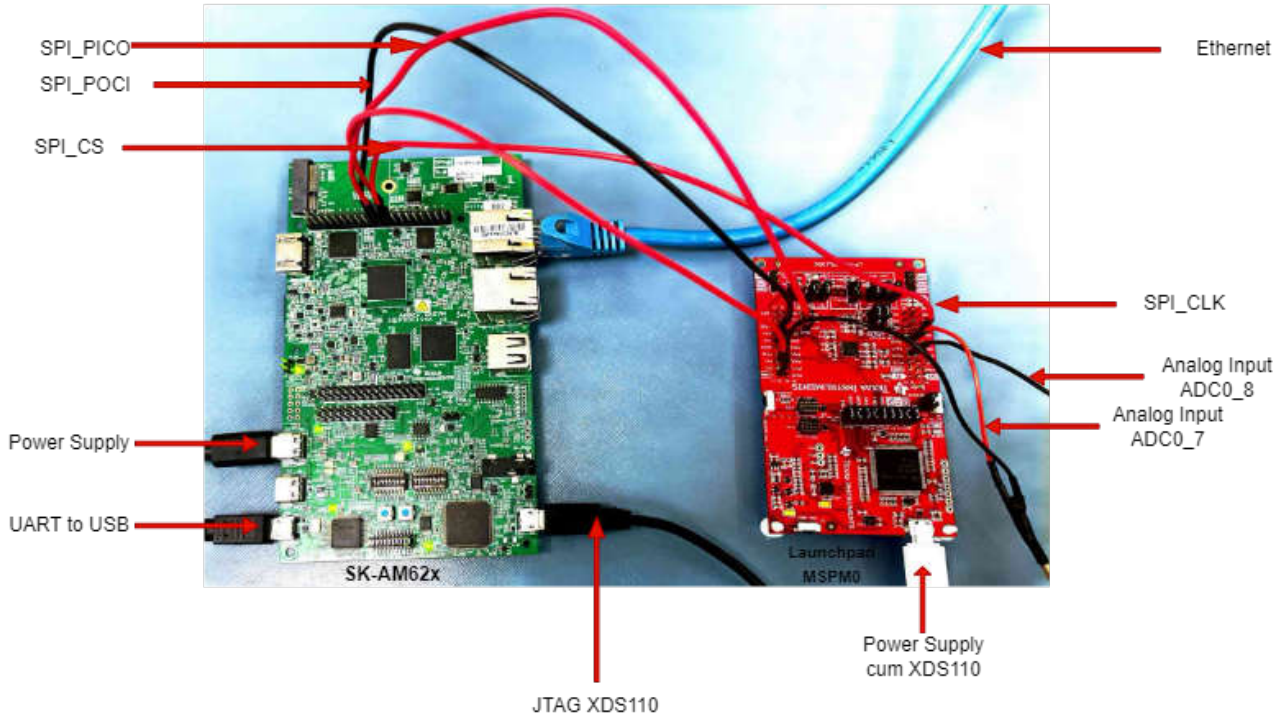


图 2-1. SK-AM62x（A53 内核）和 LP-MSPM0L1306 之间用于 SPI 通信的电缆连接。

- 对于 SK-AM62x：
  - 连接 Type-C 电源。
  - 将 JTAG XDS110 的 UART 转 USB 和 USB 连接到您的计算机。
- 对于 LP-MSPM0：
  - 将电源和 XDS110 连接到您的计算机。
  - 将模拟信号输入连接到 LaunchPad MSPM0 中的 J3_PA18 (ADC0_7)。
  - 如果需要，将模拟信号输入连接到 LaunchPad MSPM0 中的 J3_PA16 (ADC0_8)。
- 对于 SK-AM62x 到 LP-MSPM0 的连接
  - 将 SK-AM62x 用户扩展连接器中的引脚 19（B13：SPI0_D0）连接到 LaunchPad MSPM0 中的 J2_PA4 (SPI_POCI)。
  - 将 SK-AM62x 用户扩展连接器中的引脚 21（B14：SPI0_D1）连接到 LaunchPad MSPM0 中的 J2_PA5 (SPI_PICO)。
  - 将 SK-AM62x 用户扩展连接器中的引脚 23（A14：SPI0_CLK）连接到 LaunchPad MSPM0 中的 J1_PA6 (SPI_CLK)。
  - 将 SK-AM62x 用户扩展连接器中的引脚 24（A13：SPI0_CS0）连接到 LaunchPad MSPM0 中的 J2_PA3 (SPI_CS(PWM))。

2.1.2 M4F 内核硬件设置

为了使用 M4F 内核，SK-AM62x 上 SPI 的外设引脚位于 MCU 接头中。图 2-2 显示了硬件设置。

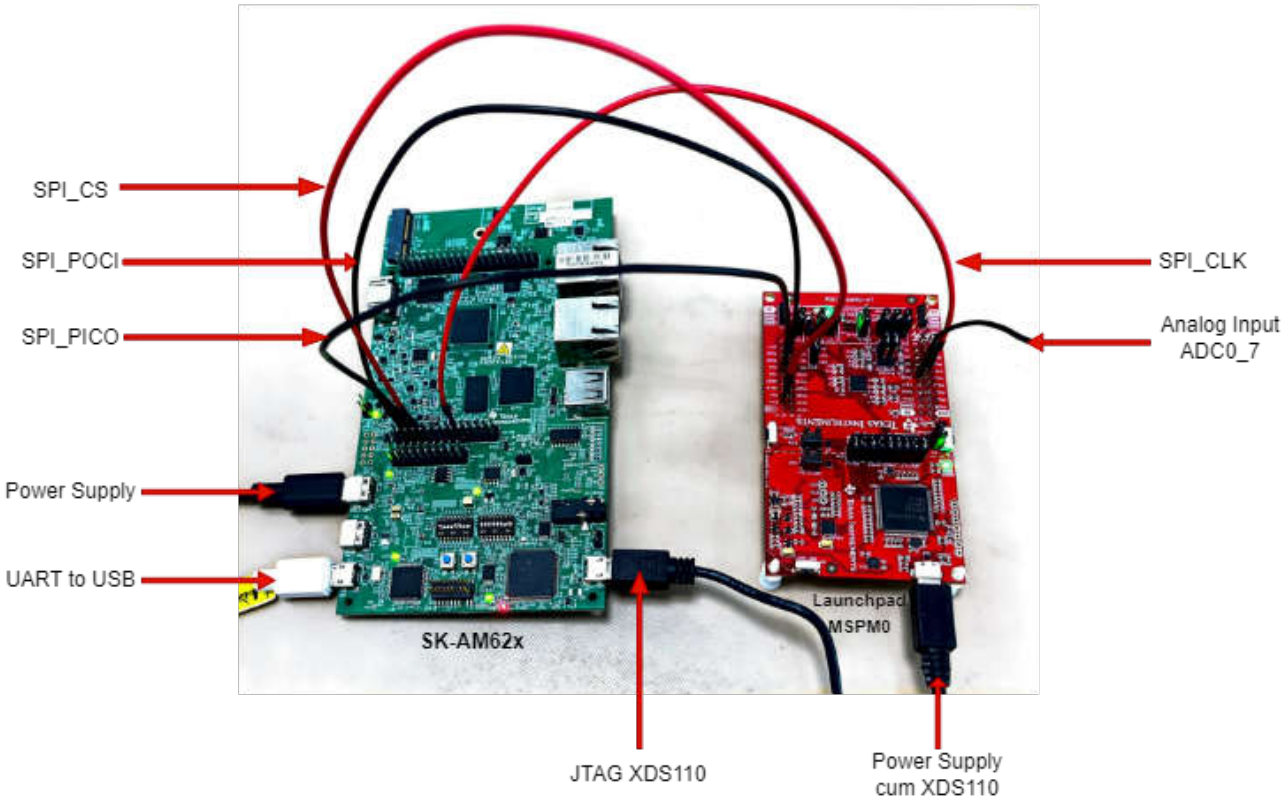


图 2-2. SK-AM62x（M4F 内核）和 LP-MSPM0L1306 之间用于 SPI 通信的电缆连接。

- 对于 SK-AM62x：
  - 连接 Type-C 电源。
  - 将 JTAG XDS110 的 UART 转 USB 和 USB 连接到您的计算机。
- 对于 LP-MSPM0：
  - 将电源和 XDS110 连接到您的计算机。
  - 将模拟信号输入连接到 LP-MSPM0 中的 J3_PA18 (ADC0_7)。
  - 如果需要，将模拟信号输入连接到 LaunchPad MSPM0 中的 J3_PA16 (ADC0_8)。
- 对于 SK-AM62x 到 LP-MSPM0 的连接：
  - 将 SK-AM62x MCU 接头中的（C9：MCU_SPI0_D1）连接到 LP-MSPM0 中的 J2_PA4 (SPI_POCI)。
  - 将 SK-AM62x MCU 接头中的（D9：MCU_SPI0_D0）连接到 LP-MSPM0 中的 J2_PA5 (SPI_PICO)。
  - 将 SK-AM62x MCU 接头中的（B8：MCU_SPI0_CS1）连接到 LP-MSPM0 中的 J2_PA3 (SPI_CS(PWM))。
  - 将 SK-AM62x MCU 接头中的（A7：MCU_SPI0_CLK）连接到 LP-MSPM0 中的 J1_PA6 (SPI_CLK)。

2.2 AM62L

AM62L 只有 1 个域 - 具有 2 个 A53 内核的 MAIN 域，并在其中实现了通过 ADC 进行 SPI 数据传输的硬件设置。硬件设置如下所示。

2.2.1 A53 内核硬件设置

为了使用 A53 内核，AM62L 上的 SPI 外设引脚位于用户扩展接头中。图 2-3 显示了硬件设置。

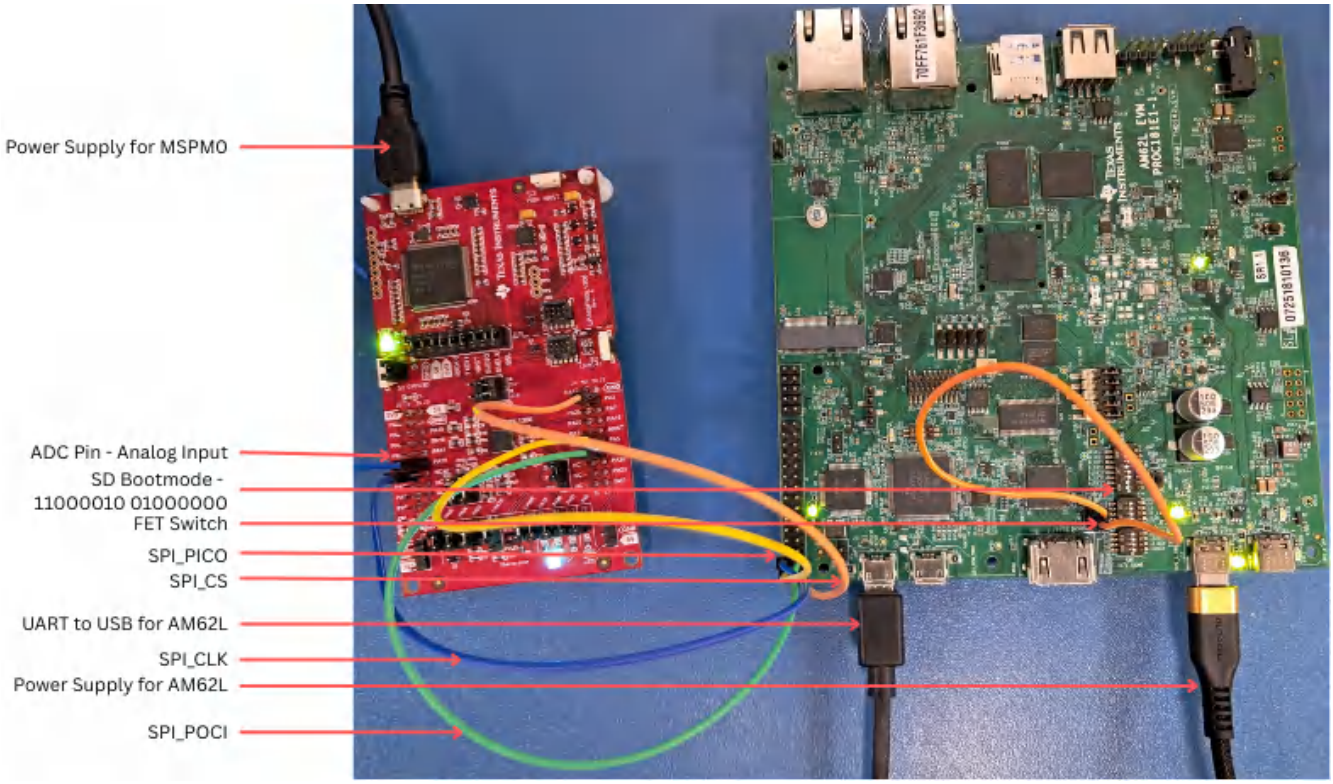


图 2-3. AM62L（A53 内核）和 LP-MSPM0L1306 之间用于 SPI 通信的电缆连接。

- 对于 AM62L：
  - 连接 Type-C 电源
  - 将 JTAG XDS110 的 UART 转 USB 和 USB 连接到您的计算机。
- 对于 LP-MSPM0：
  - 将电源和 XDS110 连接到您的计算机。
  - 将模拟信号输入连接到 LP-MSPM0 中的 J3_PA18 (ADC0_7)。
  - 如果需要，将模拟信号输入连接到 LaunchPad MSPM0 中的 J3_PA16 (ADC0_8)。
- 用于 AM62L EVM 与 MSPM0L 的连接
  - 将 AM62L EVM 用户扩展连接器中的引脚 28 (SPI1_D0_EXP) 连接到 Launchpad MSPM0Lx 中的 J2_PA4 (SPI_POCI)。
  - 将 AM62L EVM 用户扩展连接器中的引脚 29 (SPI1_D1_EXP) 连接到 Launchpad MSPM0Lx 中的 J2_PA5 (SPI_PICO)。
  - 将 AM62L EVM 用户扩展连接器中的引脚 27 (SPI1_CLK_EXP) 连接到 Launchpad MSPM0Lx 中的 J1_PA6 (SPI_CLK)。
  - 将 AM62L EVM 用户扩展连接器中的引脚 30 (SPI1_CS0_EXP) 连接到 Launchpad MSPM0Lx 中的 J2_PA3 (SPI_CS(PWM))。
- TMDS62LEVM FET 开关 J29 控制连接到 HDMI 或 IO 扩展器的 SOC 输出，默认为 HDMI。因此，需要对其进行切换。这种配置可确保在引导时，地址 0x23 处的 I2C 扩展器的引脚 1 自动设置为高输出状态。

3 软件设置

以下是设置 AM62x 或 AM62L 以及 MSPM0L1306 的步骤。

3.1 克隆 Beyond SDK GitHub 存储库

- Beyond SDK 是一个 GitHub 存储库，其中包含本实验所需的部分文件
- 在 *Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach* 中查找这些文件
- 选择控制器 - AM62x 或 AM62L
- 根据应用选择其中一个文件夹 (x_Byte_x_Channel_SPI)
- 对于 AM62x 控制器，A53 内核文件夹包含 C 文件，M4 内核包含 CCS 工程
- 在外设中，有一个适用于 MSPM0 的 CCS 工程
- 克隆 GitHub 存储库的方法如下：

```
HOST$ mkdir <Beyond-SDK-installation-path>
HOST$ cd <Beyond-SDK-installation-path>
HOST$ git clone https://github.com/TexasInstruments/Beyond-SDK.git
```

3.2 SK-AM62x 软件设置

3.2.1 A53 内核

- 遵循 AM62x 入门套件 EVM 快速入门指南中提供的设置。
- 以下实验中使用了适用于 AM62x 的 Processor SDK Linux 版本 9.0。
- 通过按以下步骤修改内核器件树，在 Linux 中设置 SPI 驱动程序：

1. 在路径 *<psdk-installation-path>/board-support/ti-linux-kernel/arch/arm64/boot/dts/ti* 上找到 k3-am625-sk.dts 器件树文件
2. 按如下方式修改该文件：

在 &main_pmx0{...} 内添加：

```
main_spi0_pins_default: main-spi0-pins-default {
        pinctrl-single,pins = <
             AM62X_IOPAD(0x01bc, PIN_OUTPUT, 0) /* (A14) SPI0_CLK */
             AM62X_IOPAD(0x01c0, PIN_INPUT, 0) /* (B13) SPI0_D0 */
             AM62X_IOPAD(0x01c4, PIN_OUTPUT, 0) /* (B14) SPI0_D1 */
             AM62X_IOPAD(0x01b4, PIN_OUTPUT, 0) /* (A13) SPI0_CS0 */
        >;
};
```

在该文件末尾添加：

```
&main_spi0 {
     status = "okay";
     pinctrl-names = "default";
     pinctrl-0 = <&main_spi0_pins_default>;
     spidev@0 {
          spi-max-frequency = <16000000>;
          reg = <0>;
          compatible = "rohm,dh2228fv";
     };
};
```

- 按照用户指南 - Processor SDK AM62x 中给出的步骤重新编译内核。按照此页面上的步骤操作时，请根据 SPI 内核驱动程序中提供的“内核配置”部分使用 *menuconfig* 自定义内核。有关更多详细信息，请参阅下方的步骤。

```
HOST$ cd <psdk-installaion-path>/board-support/ti-linux-kernel/
HOST$ make defconfig ti_arm64_prone.config
HOST$ make ARCH=arm64 menuconfig
      Device Drivers --->
          [*] SPI support
```

```
              <*> User mode SPI device driver support
#Save these changes to the .config file
HOST$ make Image dtbs modules
HOST$ sudo cp ./arch/arm64/boot/Image /media/<USER>/root/boot/
HOST$ sudo cp ./arch/arm64/boot/dts/ti/k3-am625-sk.dtb /media/root/boot/dtb/ti
HOST$ sudo -E env “PATH=$PATH” INSTALL_MOD_PATH=/media/<USER>/root make modules_install
HOST$ sync; sync
```

- 将目标 C 文件复制到 SDK 路径，并使用编译示例 Hello World 程序中给出的方法编译 C 工程文件。
- 将 SD 卡重新插入 SK-AM62x 并重新启动器件。

3.2.2 M4F 内核

- 为 AM62x 执行此处所述的 CCS (Code Composer Studio) 设置：AM62x 的入门步骤。确保在 CCS 安装期间在“Select Components”窗口中选择“MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs”。
- 将 CCS 工程导入 Project Explorer (File > Import > Code Composer Studio > CCS Projects)。在以下路径中查找 CCS 工程： *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/am62x/<x_Byte_x_Channel_SPI>/Controller/AM62x-M4F_Core_MCU_Domain/*
- 有关更一般的 CCS 支持，请参阅 CCS 用户指南。

3.3 AM62L 软件设置

3.3.1 A53 内核

- 按照 AM62L 入门指南中提供的步骤操作。
- 以下实验中使用了适用于 AM62L 的处理器 SDK Linux 版本 11.0.15。
- 通过在 AM62L SDK 的 ti-linux-kernel 存储库中应用以下补丁，在 Linux 中设置 SPI 驱动程序。
- 注意：下面提供的补丁中禁用了 HDMI。这是因为在 TMDS62LEVM 上，HDMI 与扩展接头进行多路复用，一次只能使用其中一个。

```
diff --git a/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts b/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
index 2dd056ce0538..6cf837274c6e 100644
--- a/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
+++ b/arch/arm64/boot/dts/ti/k3-am62l3-evm.dts
@@ -36,6 +36,7 @@ memory@80000000 {

     hdmi0: connector-hdmi {
         compatible = "hdmi-connector";
+        status = "disabled";
         label = "hdmi";
         type = "a";

@@ -389,6 +390,15 @@ AM62PX_IOPAD(0x0188, PIN_INPUT, 0) /* (A9) MCASP0_AXR1 */
         >;
     };

+    main_spi1_pins_default: main-spi1-pins-default {
+        pinctrl-single,pins = <
+            AM62LX_IOPAD(0x008c, PIN_OUTPUT, 4) /* (H22) SPI1_CLK */
+            AM62LX_IOPAD(0x0080, PIN_INPUT, 4) /* (K22) SPI1_D0 */
+            AM62LX_IOPAD(0x0084, PIN_OUTPUT, 4) /* (J23) SPI1_D1 */
+            AM62LX_IOPAD(0x0088, PIN_OUTPUT, 4) /* (K23) SPI1_CS0 */
+        >;
+    };
+
     pmic_irq_pins_default: pmic-irq-default-pins {
         pinctrl-single,pins = <
             AM62LX_IOPAD(0x01e8, PIN_INPUT, 0) /* (C8) EXTINTn */
@@ -410,6 +420,17 @@ &main_uart0 {
     bootph-all;
 };

+&main_spi1 {
+    status = "okay";
+    pinctrl-names = "default";
+    pinctrl-0 = <&main_spi1_pins_default>;
+    spidev@0 {
+        spi-max-frequency = <24000000>;
```

```
+        reg = <0>;
+        compatible = "rohm,dh2228fv";
+    };
+};
+
 &main_uart1 {
     pinctrl-names = "default";
     pinctrl-0 = <&main_uart1_pins_default>;
@@ -488,11 +509,14 @@ exp2: gpio@23 {

     sii9022: bridge-hdmi@3b {
         compatible = "sil,sii9022";
+        status="disabled";
         reg = <0x3b>;
         interrupt-parent = <&exp1>;
         interrupts = <16 IRQ_TYPE_EDGE_FALLING>;
         #sound-dai-cells = <0>;
         sil,i2s-data-lanes = < 0 >;
+        pinctrl-names = "default";
+        pinctrl-0 = <&main_dpi_pins_default>;
         bootph-all;

         ports {
@@ -774,8 +798,6 @@ partition@7fe0000 {

 &dss {
     status = "okay";
-    pinctrl-names = "default";
-    pinctrl-0 = <&main_dpi_pins_default>;
     bootph-all;
 };

diff --git a/arch/arm64/configs/defconfig b/arch/arm64/configs/defconfig
index 94dfc265a61c..157d0d62fb53 100644
--- a/arch/arm64/configs/defconfig
+++ b/arch/arm64/configs/defconfig
@@ -1828,3 +1828,5 @@ CONFIG_CORESIGHT_STM=m
 CONFIG_CORESIGHT_CPU_DEBUG=m
 CONFIG_CORESIGHT_CTI=m
 CONFIG_MEMTEST=y
+CONFIG_UHID=y
+CONFIG_SPI_SPIDEV=y
\ No newline at end of file
```

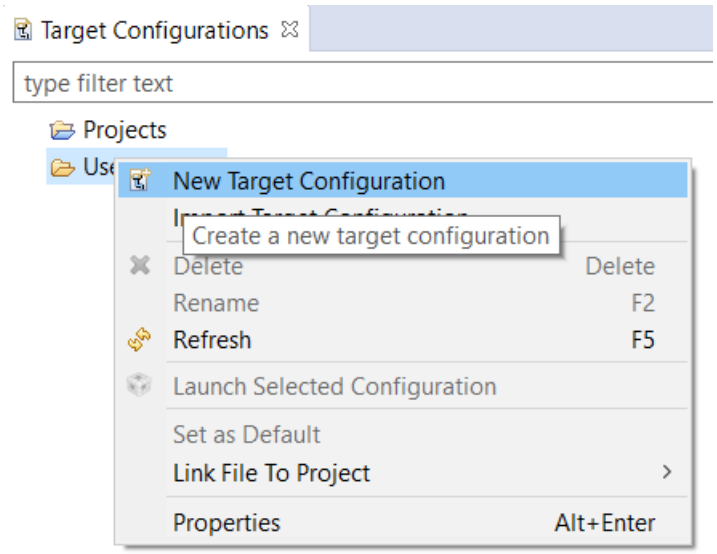
- 重新编译 Linux 并在 SD/eMMC 上安装镜像、dtb 和模块。
- 在 Linux 用户空间中使用并编译此处链接的 C 文件
- 对于多字节示例，有 2 种在控制器端接收数据的方法。一种方法是以 2 个时钟（每个时钟 8 位，来自 MSPM0）发送数据。另一种方法是在双持续时间的单个时钟突发中发送整个数据。这两种方法都已经过测试，可以从此处链接的 C 文件实现。
- **注意：**如果在 Linux 用户空间中难以在更高的 SPI 频率下运行 SPI 脚本，请连接 MSPM0 和 AM62L 的接地引脚。

3.4 LP-MSPM0L130x 软件设置

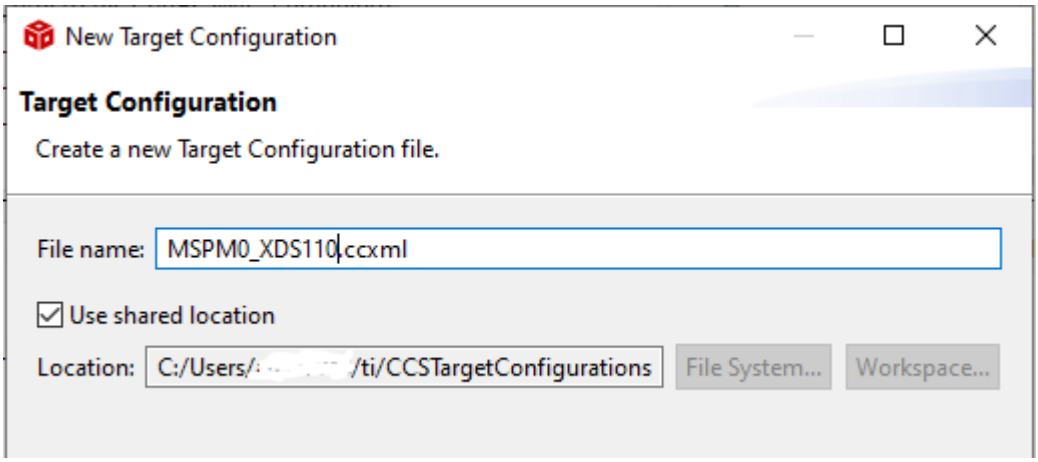
CCS (Code Composer Studio) 可用于在 MSPM0 LaunchPad 上进行开发。查看 CCS 用户指南，获取与 CCS 相关的一般帮助。要在 MSPM0 器件上进行开发，请在 CCS 安装期间在“Select Components”窗口中选择“MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs”。

按照以下步骤为 MSPM0 添加新的目标配置：

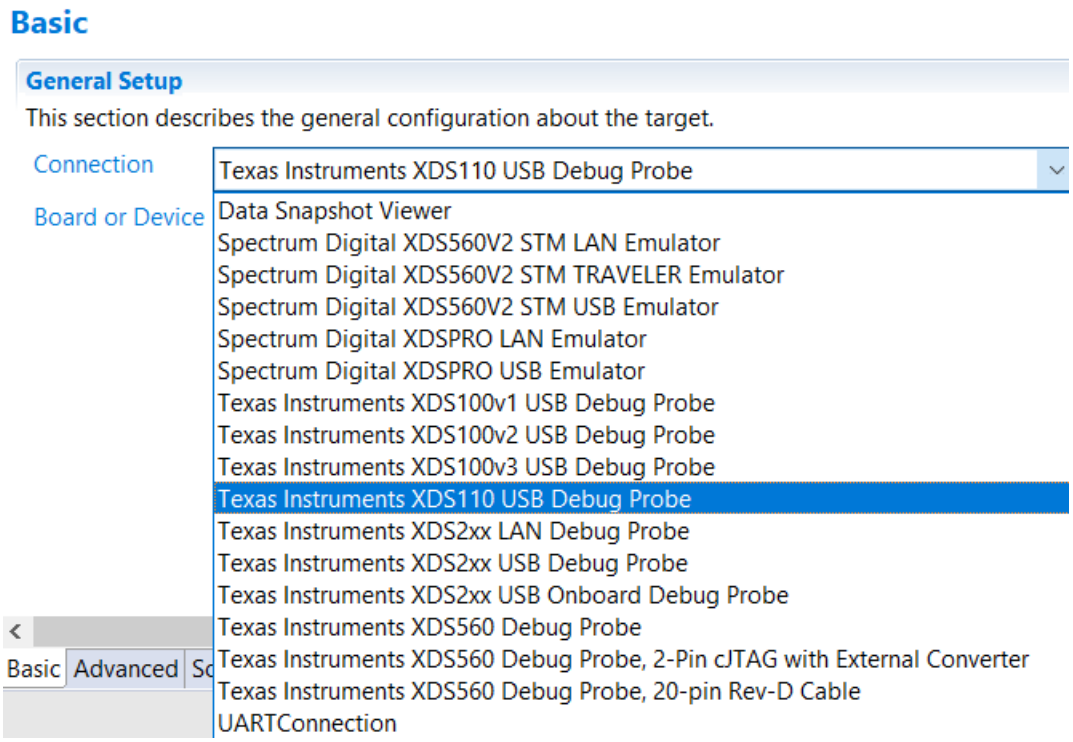
- 创建新的目标配置



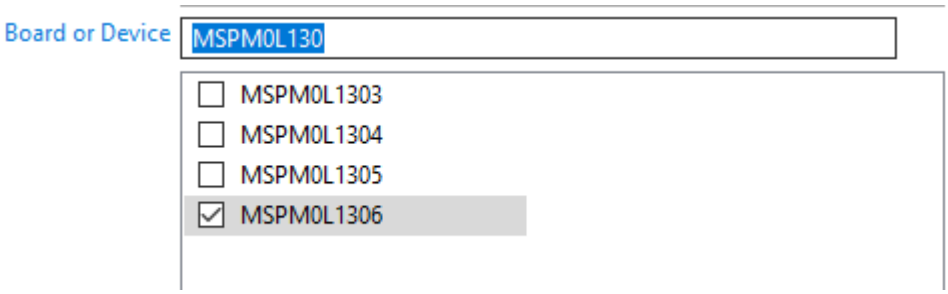
- 为新目标配置起个好名字，通常为 {soc name}_{JTAG type}



- 选择 XDS110 USB Debug Probe 作为连接



- 在“Board or Device”中，键入“MSPM0L130”，然后选择“MSPM0L1306”



- 点击“Save”以保存新创建的目标配置。

4 执行步骤

本节将讨论在控制器和外设上执行工程的步骤。

4.1 在 LP-MSPM0L130x 上运行工程

创建目标配置后，MSPM0 二进制文件可以编译并写入片上闪存：

1. 将 CCS 工程导入工作区中。这是该工程的路径：
   a. *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/<controller>/<x_Byte_x_Channel_SPI>/Peripheral/MSPM0/*
2. 构建导入的 CCS 工程。
   a. 如果存在**重复的 GPIO 引脚名称**错误，请执行以下操作：
      i. 打开 .syscfg 文件扩展名指示的系统配置文件
      ii. 打开 ADC12 配置
      iii. 转到“Pin Configuration”部分并打开 ADC12 通道 7 引脚
      iv. 将引脚名称从“ti_driverlib_gpio_GPIOPinGeneric0”更改为“ti_driverlib_gpio_GPIOPinGeneric6”
      v. 保存更改并重新构建工程
3. 可以在 *<Beyond-SDK-installation-path>/…/MSPM0/Debug/<file-name>.out* 中找到该编译二进制文件
4. 在“Target Configurations”窗口中右键点击 *MSPM0_XDS110.ccxml*
5. 选择“Launch Selected Configuration”
6. 在“Debug”窗口中，点击“Texas Instruments XDS110 USB Debug Probe_0/CORTEX_M0P”
7. 依次选择“Run”-> “Connect Target”
8. 依次选择“Run”-> “Reset”-> “Subsystem Reset”
9. 依次选择“Run”-> “Load”-> “Load Program”
10. 浏览至 MSPM0 工程的二进制文件，然后点击“OK”。
11. 这将向闪存写入二进制文件。
12. 依次选择“Run”-> “Resume”

请注意，在较新的 MSPM0 SDK 版本上，这可能会产生编译错误。若要解决此问题，请使用 CCS 中的 Resource Explorer 导入全新的 spi_peripheral_echo_interrupts_LP_MSPM0L1306_nortos_ticlang 工程，然后：

1. 将 spi_peripheral_echo_interrupts.c 从 Beyond SDK 复制到新工程。
2. 针对新工程中的 ADC12、SPI、计时器和事件，按 Beyond SDK 中的方式复制 SysConfig 设置。

有关 CCS 的其他支持，请参阅 Code Composer Studio 用户指南。

4.2 在 SK-AM62x/AM62L EVM 上运行工程

本节根据用作控制器的内核提供了执行步骤。请注意，AM62L EVM 可以遵循相同的 A53 内核步骤。

4.2.1 A53 内核

在通过 AM62x 入门套件 EVM 快速入门指南和 AM62L 快速入门指南获得的串行监视器上，转到可执行文件的位置并使用以下命令运行该可执行文件：

```
root@am62xx-evm:~#./<executable_name> -D <spidriver_name_from_/dev_folder> -s <speed> -v
```

示例：

```
root@am62xx-evm:~#./spidev_adc_multibyte_multichannel -D /dev/spidev1.0 -s 16000000 -v
```

4.2.2 M4F 内核

将预编译的 SK-AM62x 二进制文件写入片上闪存：

1. 将 CCS 工程导入到不同于 MSPM0 工作区的工作区中
   a. *<Beyond-SDK-installation-path>/Beyond-SDK/collaterals/appnotes/slaaej0-MSPM0-ADC-Attach/am62x/<x_Byte_x_Channel_SPI>/Controller/AM62x-M4F_Core_MCU_Domain/Debug/<file-name>.out*
   b. 这样就可以让每个器件有两个不同的调试会话
2. 构建导入的 CCS 工程。
3. 在“Target Configurations”窗口中右键点击 AM62x_XDS110.ccxml
4. 选择“Launch Selected Configuration”
5. 在“Debug”窗口中，点击“Texas Instruments XDS110 USB Debug Probe_0/BLAZAR_Cortex_M4F_1”
6. 依次选择“Run”->“Connect Target”
7. 依次选择“Run”->“Reset”->“Subsystem Reset”
8. 依次选择“Run”->“Load”->“Load Program”
9. 浏览至 AM62x 工程的预编译二进制文件，然后点击“OK”。
10. 这将向闪存写入二进制文件。
11. 依次选择“Run”->“Resume”

5 结果

在所用的示例应用程序代码中：

- *单字节* 是指 8 位 ADC 数据传输。
- *多字节* 是指 SPI 传输能够传输所配置的尽可能多的字节。在示例中，已配置 2 字节数据。在所传输的 16 位中，只有低 12 位包含 ADC 数据，因为 MSPM0 上 ADC 的最大分辨率为 12 位。
- *单通道* 是指 ADC 仅监控 1 个模拟信号。控制器必须发送虚拟命令来启动事务，但不需要在外设处检查命令值。
- *多通道* 是指 ADC 依次转换多个模拟信号。控制器必须发送有效命令才能启动事务并接收相应的通道数据。
- AM62x 和 AM62L 控制器提供类似的输出。

5.1 单字节单通道

请注意，为了获得如下结果，我们考虑了以下用于 8 位 ADC 的模拟输入。

命令 0x00：ADC 通道 7：正弦信号（3.3Vpp，1.65V 直流失调电压，@2Hz）

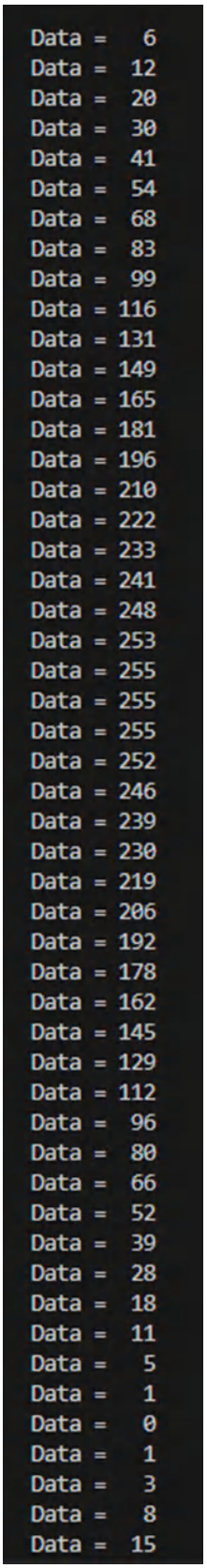


图 5-1. 单字节单通道正弦波结果

命令 0x00：ADC 通道 7：方波信号（3.3Vpp，1.65V 直流失调电压，@2Hz，50% 占空比）

```
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data =   0
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
Data = 255
```

图 5-2. 单字节单通道方波结果

5.2 单字节多通道

请注意，为了获得如下结果，我们考虑了以下用于 8 位 ADC 的模拟输入：

命令 0x00：ADC 通道 7：正弦信号（3.3Vpp，1.65V 直流失调电压，@2Hz）

命令 0x01：ADC 通道 8：直流信号 (3.3V)

```
CommandID =  0, Data = 102        CommandID =  1, Data = 255
CommandID =  0, Data =  72        CommandID =  1, Data = 255
CommandID =  0, Data =  44        CommandID =  1, Data = 255
CommandID =  0, Data =  23        CommandID =  1, Data = 255
CommandID =  0, Data =   8        CommandID =  1, Data = 255
CommandID =  0, Data =   0        CommandID =  1, Data = 255
CommandID =  0, Data =   2        CommandID =  1, Data = 255
CommandID =  0, Data =  11        CommandID =  1, Data = 255
CommandID =  0, Data =  29        CommandID =  1, Data = 255
CommandID =  0, Data =  52        CommandID =  1, Data = 255
CommandID =  0, Data =  81        CommandID =  1, Data = 255
CommandID =  0, Data = 112        CommandID =  1, Data = 255
CommandID =  0, Data = 146        CommandID =  1, Data = 255
CommandID =  0, Data = 177        CommandID =  1, Data = 255
CommandID =  0, Data = 206        CommandID =  1, Data = 255
CommandID =  0, Data = 229        CommandID =  1, Data = 255
CommandID =  0, Data = 246        CommandID =  1, Data = 255
CommandID =  0, Data = 255        CommandID =  1, Data = 255
CommandID =  0, Data = 255        CommandID =  1, Data = 255
CommandID =  0, Data = 248        CommandID =  1, Data = 255
CommandID =  0, Data = 233        CommandID =  1, Data = 255
CommandID =  0, Data = 210        CommandID =  1, Data = 255
CommandID =  0, Data = 183        CommandID =  1, Data = 255
CommandID =  0, Data = 151        CommandID =  1, Data = 255
CommandID =  0, Data = 119        CommandID =  1, Data = 255
CommandID =  0, Data =  87        CommandID =  1, Data = 255
CommandID =  0, Data =  57        CommandID =  1, Data = 255
CommandID =  0, Data =  33        CommandID =  1, Data = 255
CommandID =  0, Data =  14        CommandID =  1, Data = 255
CommandID =  0, Data =   3        CommandID =  1, Data = 255
CommandID =  0, Data =   0        CommandID =  1, Data = 255
CommandID =  0, Data =   6        CommandID =  1, Data = 255
CommandID =  0, Data =  19        CommandID =  1, Data = 255
CommandID =  0, Data =  40        CommandID =  1, Data = 255
CommandID =  0, Data =  66        CommandID =  1, Data = 255
CommandID =  0, Data =  97        CommandID =  1, Data = 255
CommandID =  0, Data = 129        CommandID =  1, Data = 255
CommandID =  0, Data = 162        CommandID =  1, Data = 255
CommandID =  0, Data = 192        CommandID =  1, Data = 255
CommandID =  0, Data = 218        CommandID =  1, Data = 255
CommandID =  0, Data = 238        CommandID =  1, Data = 255
CommandID =  0, Data = 251        CommandID =  1, Data = 255
CommandID =  0, Data = 255        CommandID =  1, Data = 255
CommandID =  0, Data = 253        CommandID =  1, Data = 255
CommandID =  0, Data = 241        CommandID =  1, Data = 255
CommandID =  0, Data = 222        CommandID =  1, Data = 255
CommandID =  0, Data = 197        CommandID =  1, Data = 255
```

图 **5-3.** 单字节多通道结果

5.3 多字节单通道

请注意，为了获得如下结果，我们考虑了以下用于 12 位 ADC 的模拟输入。

命令 0x00：ADC 通道 7：正弦信号（3.3Vpp，1.65V 直流失调电压，@2Hz）

```
Data = 3879
Data = 4061
Data = 4095
Data = 4021
Data = 3810
Data = 3487
Data = 3070
Data = 2587
Data = 2062
Data = 1540
Data = 1065
Data =  641
Data =  307
Data =   97
Data =    6
Data =   52
Data =  232
Data =  527
Data =  915
Data = 1381
Data = 1896
Data = 2422
Data = 2915
Data = 3356
Data = 3719
Data = 3970
Data = 4095
Data = 4086
Data = 3953
Data = 3694
Data = 3326
Data = 2879
Data = 2377
Data = 1857
Data = 1338
Data =  884
Data =  497
Data =  209
Data =   43
Data =   13
Data =  111
Data =  346
Data =  683
Data = 1113
Data = 1603
Data = 2120
Data = 2639
Data = 3116
Data = 3527
```

图 5-4. 多字节单通道正弦波结果

命令 0x00：ADC 通道 7：方波信号（3.3Vpp，1.65V 直流失调电压，@2Hz，50% 占空比）

```
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data =    8
Data =    9
Data =    8
Data =    7
Data =    6
Data =    6
Data =    7
Data =    8
Data =    7
Data =    3
Data =    8
Data =    6
Data =    0
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data = 4095
Data =   13
Data =    9
Data =   13
Data =    9
Data =   12
Data =    7
Data =    8
Data =    7
Data =    7
Data =    0
Data =    4
Data =   11
Data = 4095
```

图 5-5. 多字节单通道方波结果

5.4 多字节多通道

请注意，为了获得如下结果，我们考虑了以下用于 12 位 ADC 的模拟输入：

命令 0x00：ADC 通道 7：正弦信号（3.3Vpp，1.65V 直流失调电压，@2Hz）

命令 0x01：ADC 通道 8：直流信号 (3.3V)

```
CommandID =  0, Data =  2501        CommandID =  1, Data =  4094
CommandID =  0, Data =  3421        CommandID =  1, Data =  4095
CommandID =  0, Data =  3993        CommandID =  1, Data =  4092
CommandID =  0, Data =  4079        CommandID =  1, Data =  4088
CommandID =  0, Data =  3657        CommandID =  1, Data =  4095
CommandID =  0, Data =  2822        CommandID =  1, Data =  4091
CommandID =  0, Data =  1798        CommandID =  1, Data =  4095
CommandID =  0, Data =   840        CommandID =  1, Data =  4095
CommandID =  0, Data =   191        CommandID =  1, Data =  4095
CommandID =  0, Data =    16        CommandID =  1, Data =  4089
CommandID =  0, Data =   359        CommandID =  1, Data =  4088
CommandID =  0, Data =  1128        CommandID =  1, Data =  4095
CommandID =  0, Data =  2132        CommandID =  1, Data =  4095
CommandID =  0, Data =  3123        CommandID =  1, Data =  4090
CommandID =  0, Data =  3840        CommandID =  1, Data =  4086
CommandID =  0, Data =  4095        CommandID =  1, Data =  4093
CommandID =  0, Data =  3859        CommandID =  1, Data =  4086
CommandID =  0, Data =  3154        CommandID =  1, Data =  4086
CommandID =  0, Data =  2172        CommandID =  1, Data =  4095
CommandID =  0, Data =  1160        CommandID =  1, Data =  4088
CommandID =  0, Data =   370        CommandID =  1, Data =  4089
CommandID =  0, Data =    20        CommandID =  1, Data =  4090
CommandID =  0, Data =   172        CommandID =  1, Data =  4093
CommandID =  0, Data =   807        CommandID =  1, Data =  4092
CommandID =  0, Data =  1767        CommandID =  1, Data =  4090
CommandID =  0, Data =  2789        CommandID =  1, Data =  4094
CommandID =  0, Data =  3632        CommandID =  1, Data =  4089
CommandID =  0, Data =  4074        CommandID =  1, Data =  4095
CommandID =  0, Data =  4002        CommandID =  1, Data =  4092
CommandID =  0, Data =  3443        CommandID =  1, Data =  4090
CommandID =  0, Data =  2535        CommandID =  1, Data =  4084
CommandID =  0, Data =  1509        CommandID =  1, Data =  4090
CommandID =  0, Data =   618        CommandID =  1, Data =  4095
CommandID =  0, Data =    88        CommandID =  1, Data =  4089
CommandID =  0, Data =    63        CommandID =  1, Data =  4095
CommandID =  0, Data =   549        CommandID =  1, Data =  4087
CommandID =  0, Data =  1410        CommandID =  1, Data =  4095
CommandID =  0, Data =  2440        CommandID =  1, Data =  4089
CommandID =  0, Data =  3376        CommandID =  1, Data =  4087
CommandID =  0, Data =  3974        CommandID =  1, Data =  4092
CommandID =  0, Data =  4089        CommandID =  1, Data =  4092
```

图 5-6. 多字节多通道结果

6 总结

AM62x 和 AM62L 是各种嵌入式应用程序的理想之选。大多数嵌入式应用程序都需要从传感器收集实际的模拟信号。本文介绍了使用 MSPM0 将 8 位和 12 位 ADC 集成到 AM62x 和 AM62L 处理器所遵循的步骤。这种集成解决了一项关键限制，即 AM62x 处理器缺少任何板载 ADC 功能，而 AM62L 处理器具有约 10 位 ENOB ADC，此类 ADC 可能需要更高的 12 位分辨率来满足某些应用程序的需求。

该解决方案架构将 AM62x/AM62L 处理器定位为 SPI 控制器（使用 A53 或 M4F 内核），并将 MSPM0L130x 微控制器定位为外设。通过全双工 SPI 进行通信，控制器的速度高达 50MHz，但 MSPM0L 外设的速度限制为 16MB/s。该系统支持多种工作模式，包括单字节或多字节传输与单通道或多通道配置相结合，允许同时监控多个模拟输入。ADC 采样由计时器触发，并通过连续数据更新来保持实时性能。

硬件设置需要使用 SK-AM62x 入门套件或 AM62L EVM 通过扩展接头的 SPI 连接来连接到 LP-MSPM0L1306 LaunchPad，并需要使用模拟信号源进行测试。软件实现涉及针对 SPI 驱动程序支持的 Linux 内核修改、用于适当引脚多路复用的器件树配置、适用于 MSPM0 外设和 AM62x M4F 内核的 CCS 工程，以及用于 A53 内核 Linux 用户空间执行的编译可执行文件。该解决方案已成功使用正弦波和方波测试输入（2Hz 频率下为 3.3Vpp）在 SK-AM62x 和 AM62L 两种控制器上进行了验证，展示了在所有配置模式下均能准确采集 ADC 数据，并在支持的最高 SPI 速度下保持可靠性能。

7 参考资料

1. 德州仪器 (TI)，*AM625* 产品页面。
2. 德州仪器 (TI)，*AM62L*
3. 德州仪器 (TI)，*MSPM0L1306* 产品页面。
4. 德州仪器 (TI)，*AM625 Sitara 处理器* 数据表。
5. 德州仪器 (TI)，*AM62Lx Sitara 处理器*，数据表。
6. 德州仪器 (TI)，*MSPM0L130x 混合信号微控制器* 数据表。
7. 德州仪器 (TI)，*SK-AM62 用户指南* 用户指南。
8. 德州仪器 (TI)，*AM62L 用户指南* 用户指南。
9. 德州仪器 (TI)，*LP-MSPM0L1306 用户指南* 用户指南。
10. 德州仪器 (TI)，*SK-AM62x 快速入门指南* 快速入门指南。
11. 德州仪器 (TI)，*AM62L 快速入门指南* 快速入门指南。
12. 德州仪器 (TI)，*MSPM0 快速入门指南* 快速入门指南。
13. 德州仪器 (TI)，*内核：基础组件：处理器 SDK Linux AM62x* 用户指南。
14. 德州仪器 (TI)，*内核基础元件：处理器 SDK Linux AM62L* 用户指南。
15. 德州仪器 (TI)，*适用于 AM62x 的处理器 SDK Linux* 网页。
16. 德州仪器 (TI)，*适用于 AM62L 的 Linux SDK* 网页。
17. 德州仪器 (TI)，*编译“Hello World”程序* 网页。
18. 德州仪器 (TI)，*SK-AM62x 概述入门指南* 用户指南。
19. 德州仪器 (TI)，*AM62L 概述入门指南* 用户指南。
20. 德州仪器 (TI)，*MSPM0L1306 LaunchPad 开发套件* 用户指南。

8 Revision History

Changes from Revision A (April 2026) to Revision B (June 2026) **Page**

- 已更正排印错误 22

Changes from Revision * (November 2023) to Revision A (April 2026) **Page**

- 更新了添加 AM62L 详细信息后的摘要 1
- 添加了 AM62L 处理器图 2
- 添加了 AM62x 和 AM62L 硬件设置的基本介绍，以及有关使用 MSPM0 进行 SPI 通信的 AM62L 硬件设置信息 6
- 更新了克隆 GitHub 存储库的文件路径和步骤，更新了在 AM62x A53 内核部分中编译 Hello World 示例的链接，添加了 AM62L 软件设置的详细信息，并提供了在 AM62L 上启用 SPI 和禁用 HDMI 的补丁 9
- 更新了工程的路径以及为在 MSPM0 上运行工程构建二进制文件的步骤，添加了适用于 A53 内核的 AM62L 快速入门指南的链接，并从 Github 更新了 M4F 内核的 CCS 工程路径 14
- 根据新增的 AM62L 更新了详细信息 22
- 添加了所需的 AM62L 链接并更新了损坏的 AM62x 链接 23

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