Application Note

使用 SPI 将 MSPM0 – ADC 连接到 AM62x 上

Anshu Madwesh, Divyansh Mittal, Anil Swargam, Anshu Choudhary, and Krunal Bhargav.

摘要

本应用手册介绍了我们如何借助串行外设接口 (SPI) 将 MSPM0 上的 ADC 集成到 AM62x 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。AM62x 未随附板载 ADC，因此本文旨在演示如何将 MSPM0 微控制器的 ADC 集成到 AM62x 中。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 AM62x 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 引言

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 微处理器入门套件。这里，AM62x 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]

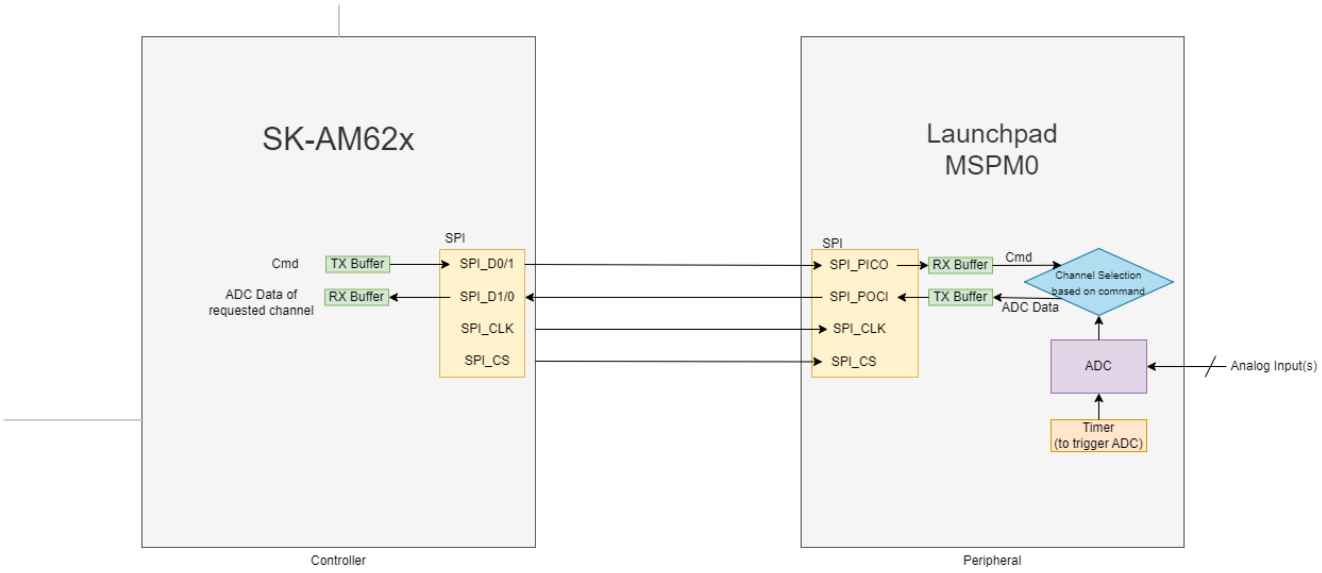


图 1-1. 控制器 (SK-AM62x) 和外设 (LP-MSPM0L130x) 之间的总体数据流

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，包括连接 SK-AM62x 和 LP-MSPM0L130x。
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 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。

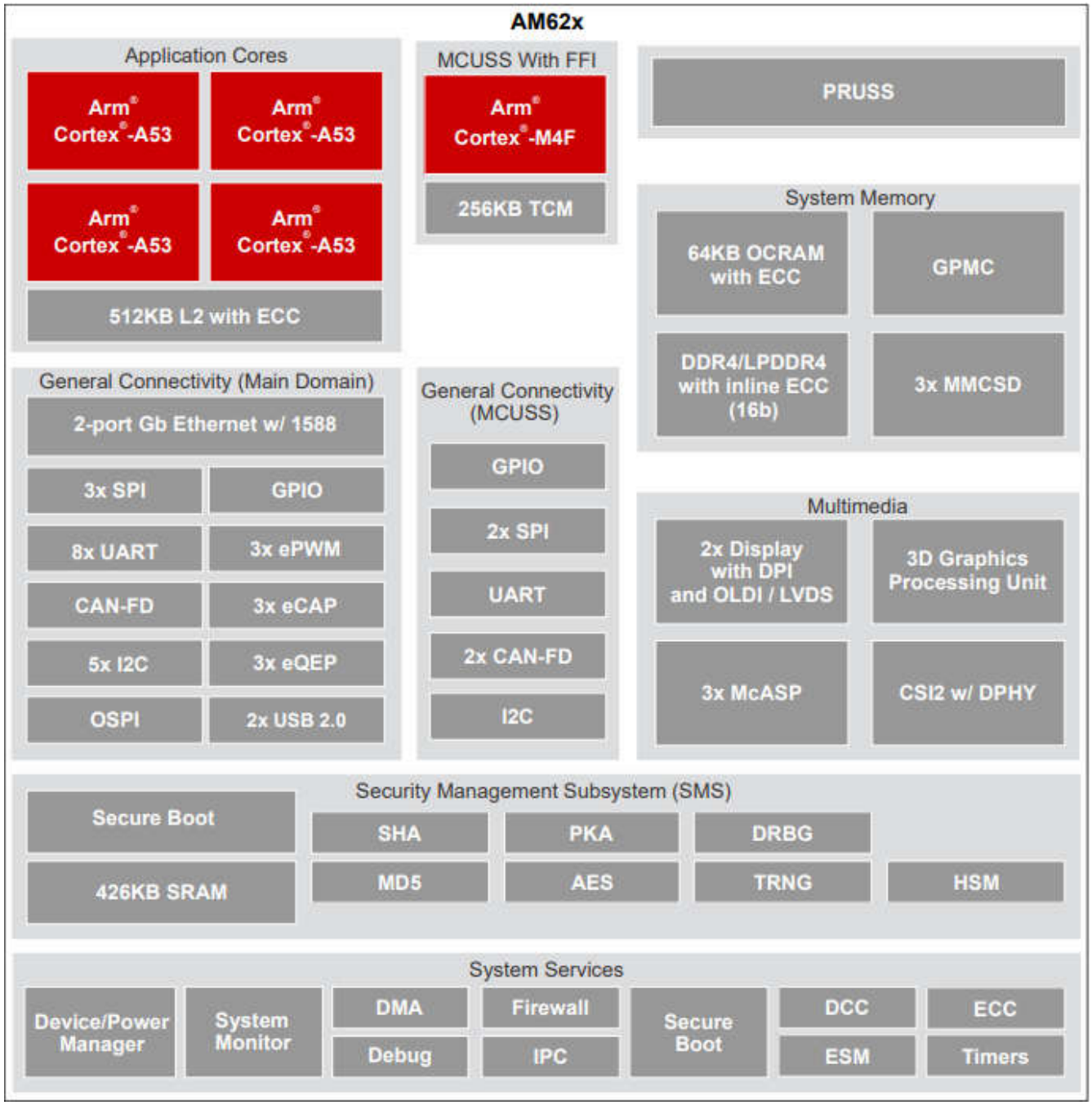


图 1-2. AM62x 简化方框图

1.3 MSPM0L130x 微控制器

MSPM0L130x 微控制器（如图 1-3 所示）是一款易于使用的评估模块 (EVM)。

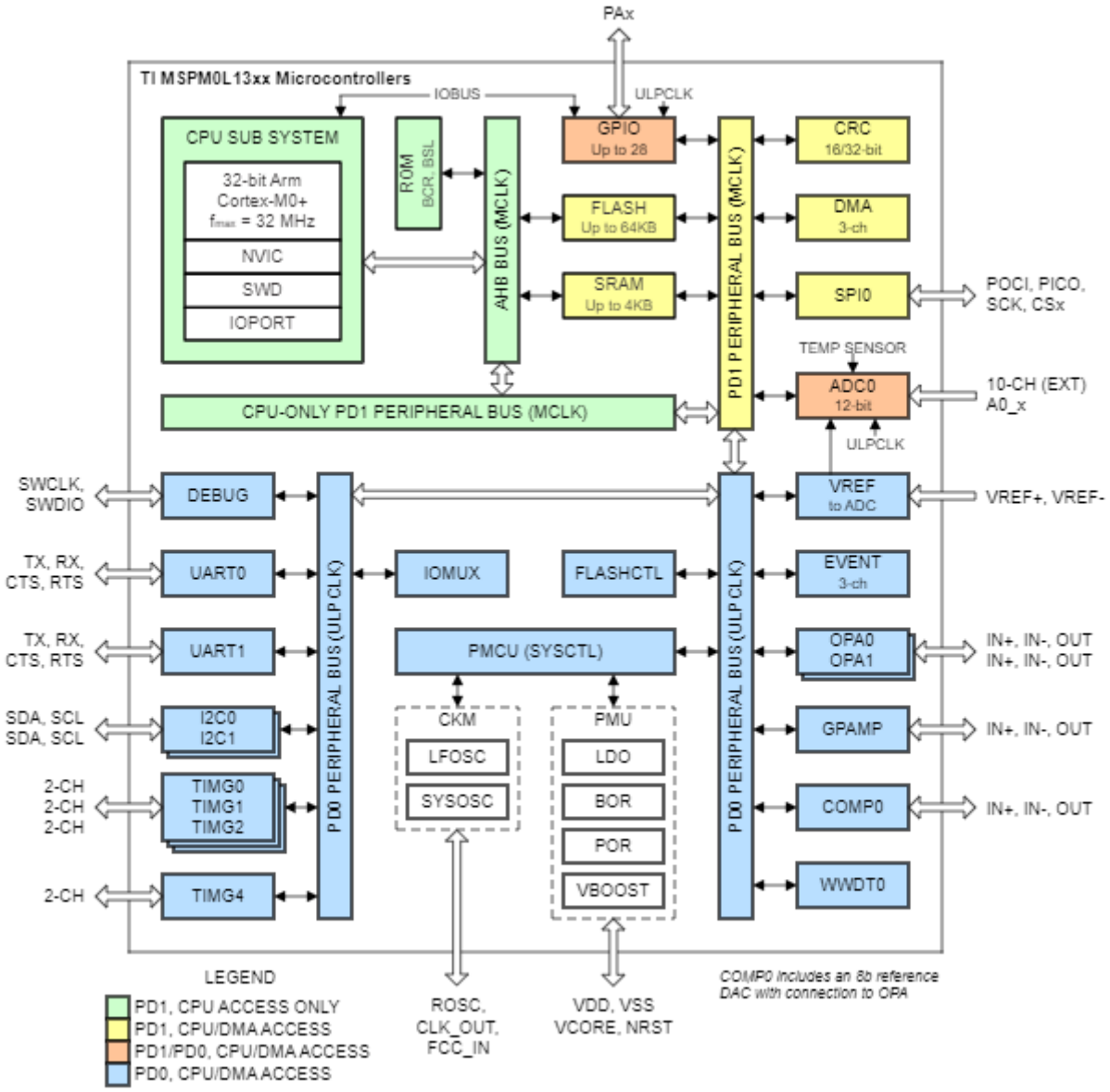


图 1-3. MSPM0L130x 简化方框图

AM62x 中来自机器视觉上下文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 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。

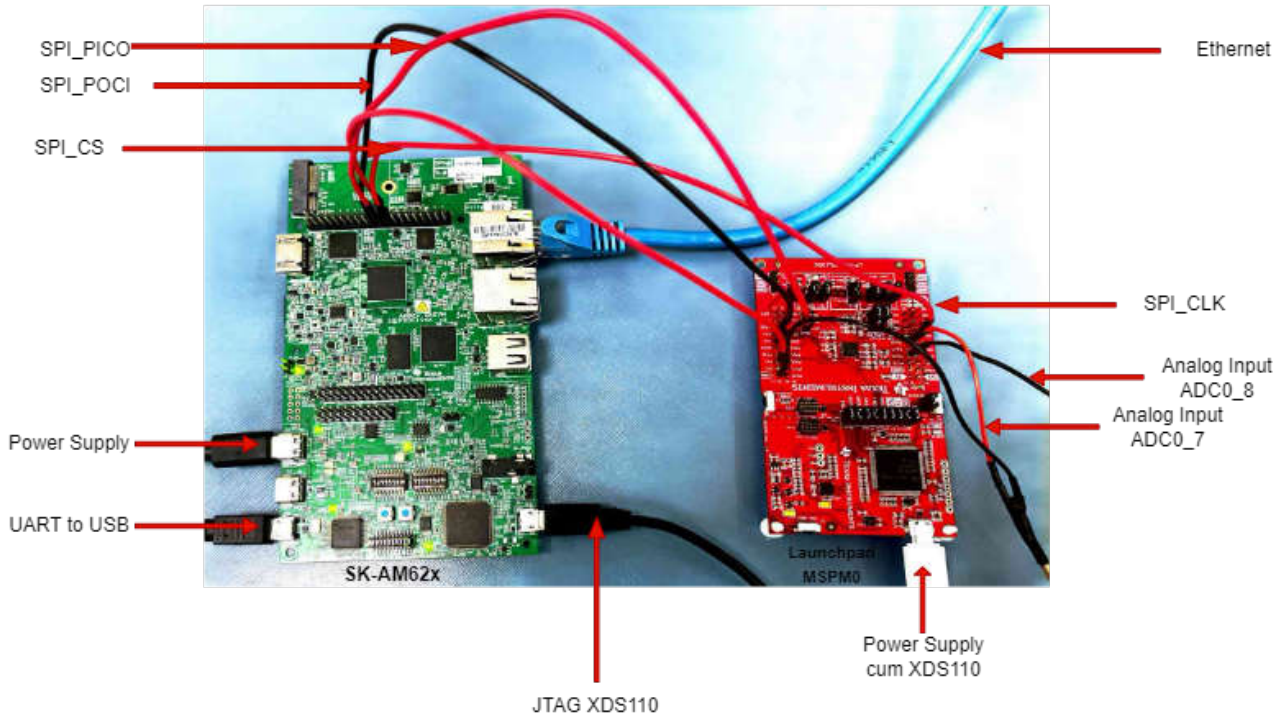


图 2-1. SK-AM62x（A53 内核）和 LP-MSPM0L1306 之间用于 SPI 通信的电缆连接。

- 对于 SK-AM62x：
  - 使用 5V 适配器连接 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)。
- 对于板间连接
  - 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))。

2.2 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。

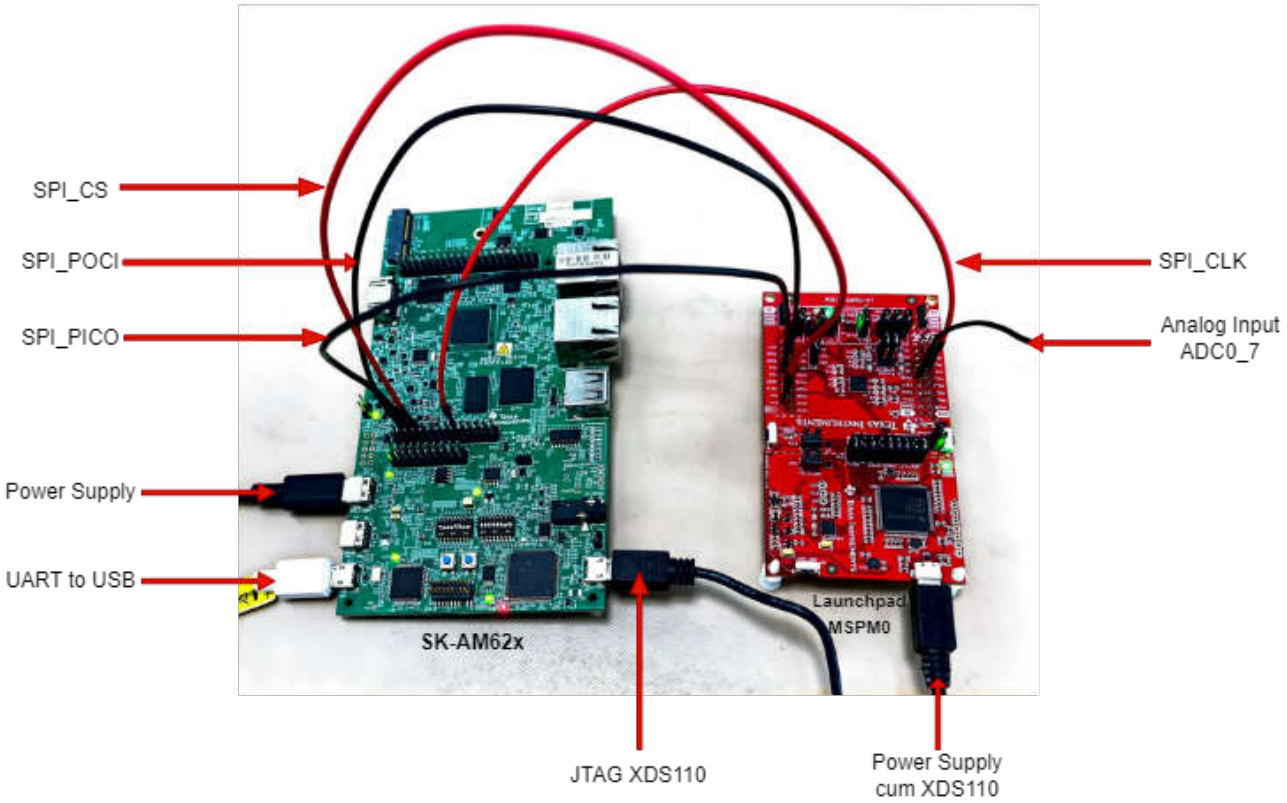


图 2-2. SK-AM62x（M4F 内核）和 LP-MSPM0L1306 之间用于 SPI 通信的电缆连接。

- 对于 SK-AM62x：
  - 使用 5V 适配器连接 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)。
- 对于板间连接：
  - 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)。

3 软件设置

以下是根据将使用的内核和 MSPM0L1306 设置 AM62x 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/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI* 中查找这些文件
- 根据应用选择其中一个文件夹 (x_Byte_x_Channel_SPI)
- 在控制器中，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 程序中给出的任何方法编译 C 工程文件。无论使用哪种方法，最后都应将可执行文件加载到 SD 卡中。有关更多详细信息，请参阅下方的步骤。

```
HOST$ cd <Beyond-SDK-installation-path>/Beyond-SDK/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/
<x_Byte_x_Channel_SPI>/Controller/AM62x-A53_Core_MAIN_Domain/
HOST$ cp <target-filename>.c <psdk-installation-path>/linux-devkit/
HOST$ cd <psdk-installation-path>/linux-devkit/
HOST$ source environment-setup
HOST$ ${CC} <target-filename>.c -o <output-filename>
HOST$ sudo cp <output-filename> /media/<USER>/root/home/root
HOST$ exit
```

- 将 SD 卡重新插入 SK-AM62x 并重新启动器件。

3.2.2 M4F 内核

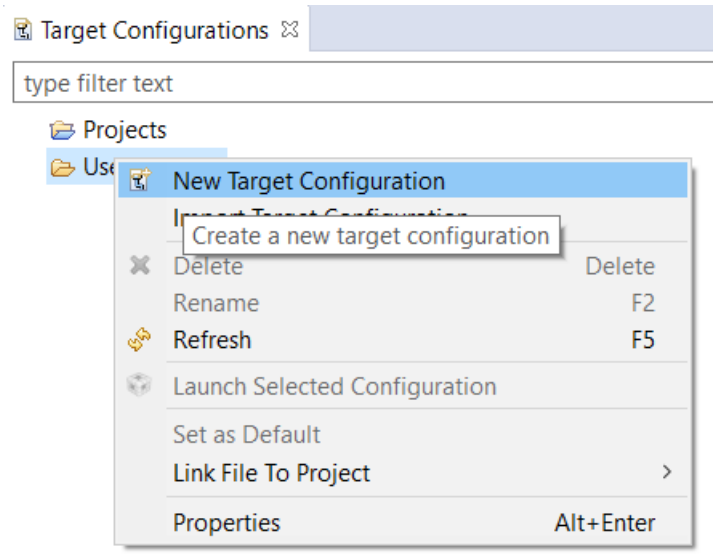
- 为 AM62x 执行此处所述的 CCS (Code Composer Studio) 设置：AM62x 的入门步骤。确保在 CCS 安装期间在“Select Components”窗口中选择“MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs”。
- 将 CCS 工程导入 Project Explorer (File > Import > Code Composer Studio > CCS Projects)。在以下路径中查找 CCS 工程：*<Beyond-SDK-installation-path>/Beyond-SDK/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/<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 LP-MSPM0L130x 软件设置

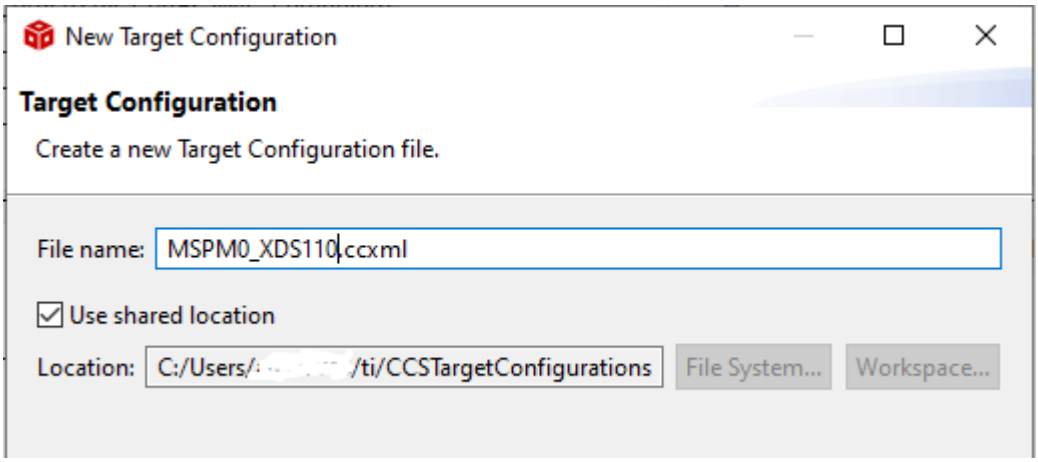
CCS (Code Composer Studio) 可用于在 MSPM0 LaunchPad 上进行开发。查看 CCS 用户指南，获取与 CCS 相关的一般帮助。要在 MSPM0 器件上进行开发，请在 CCS 安装期间在“Select Components”窗口中选择“MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs”。

按照以下步骤为 MSPM0 添加新的目标配置：

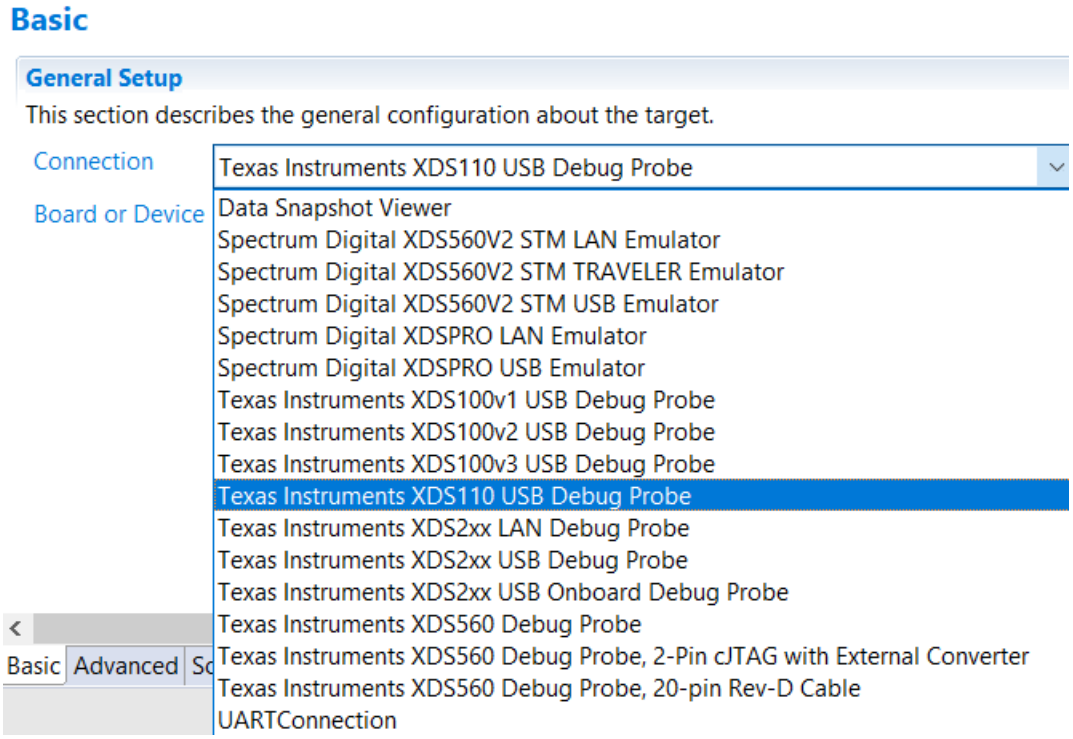
- 创建新的目标配置



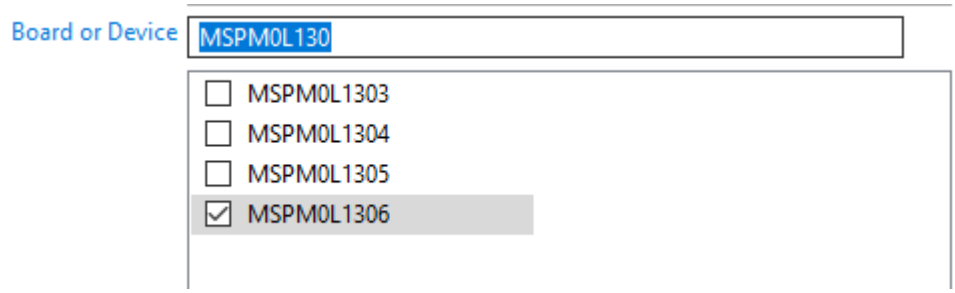
- 为新目标配置起个好名字，通常为 {soc name}_{JTAG type}



- 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，预编译的 MSPM0 二进制文件可以写入片上闪存：

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/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/<x_Byte_x_Channel_SPI>/Peripheral/MSPM0/*
   b. 可以在 *<Beyond-SDK-installation-path>/…/MSPM0/Debug/<file-name>.out* 中找到该二进制文件
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. 在“Target Configurations”窗口中右键点击 *MSPM0_XDS110.ccxml*
4. 选择“Launch Selected Configuration”
5. 在“Debug”窗口中，点击“Texas Instruments XDS110 USB Debug Probe_0/CORTEX_M0P”
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 MSPM0 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”

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

4.2 在 SK-AM62x 上运行工程

本节根据用作控制器的内核提供了执行步骤。

4.2.1 A53 内核

在通过 AM62x 入门套件 EVM 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/am62x/MSPM0-ADC-RTC-Attach/MSPM0-ADC-Attach-SPI/<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。

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）

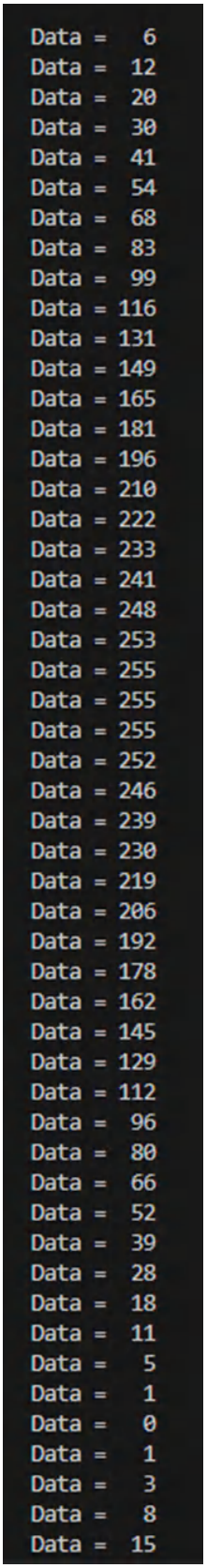


图 5-1. 单字节单通道正弦波结果

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）

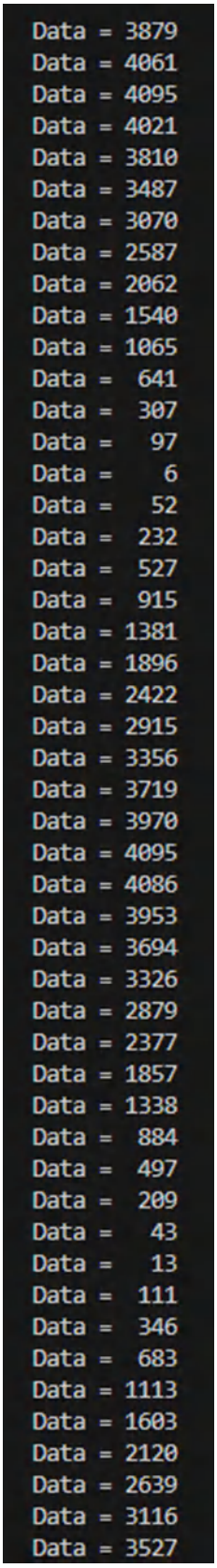


图 5-4. 多字节单通道正弦波结果

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)

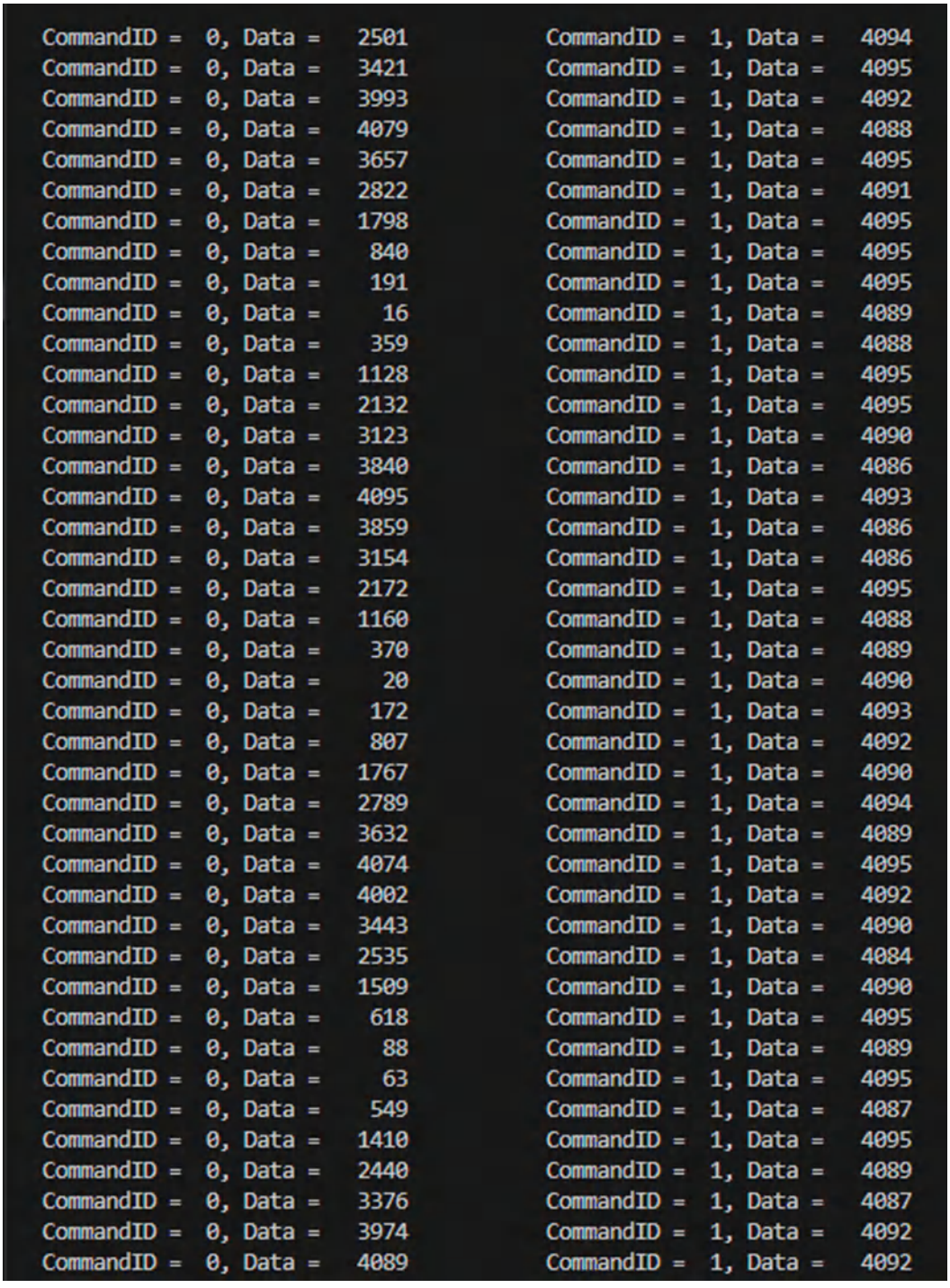


图 5-6. 多字节多通道结果

6 总结

AM62x 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。本文档介绍了将 MSPM0 上的板载 ADC 集成到 AM62x 中所遵循的步骤。该应用显示出低延迟和高达 16MHz 的 SPI 操作速度，这是通过 MSPM0L130x 可获得的最大值。

7 参考资料

1. 德州仪器 (TI)，“AM625”，[在线]。网址：https://www.ti.com.cn/product/cn/AM625
2. 德州仪器 (TI)，“MSPM0L1306”，[在线]。网址：https://www.ti.com.cn/product/cn/MSPM0L1306
3. 德州仪器 (TI)，“AM625 Sitara 处理器数据表”，[在线]。网址：https://www.ti.com/document-viewer/am625/datasheet
4. 德州仪器 (TI)，“AM625 Sitara 处理器数据表”，[在线]。网址：https://www.ti.com/document-viewer/mspm0l1306/datasheet
5. 德州仪器 (TI)，“SK-AM62 用户指南”，[在线]。网址：https://www.ti.com/document-viewer/lit/html/spruj40
6. 德州仪器 (TI)，“LP-MSPM0L1306 用户指南”，[在线]。网址：https://www.ti.com/document-viewer/lit/html/slau869
7. 德州仪器 (TI)，“SK-AM62 快速入门指南”，[在线]。网址：https://dev.ti.com/tirex/content/tirex-product-tree/am62x-devtools/docs/am62x_skevm_quick_start_guide.html
8. 德州仪器 (TI)，“内核：基础组件：Processors SDK Linux AM62x”，[在线]。网址：https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/09_00_00_03/exports/docs/linux/Foundational_Components_Kernel_Users_Guide.html
9. 德州仪器 (TI)，“SPI 内核驱动程序：基础组件：Processors SDK Linux AM62x”，[在线]。网址：https://software-dl.ti.com/processor-sdk-linux/esd/AM62X/09_00_00_03/exports/docs/linux/Foundational_Components/Kernel/Kernel_Drivers/SPI.html
10. 德州仪器 (TI)，“编译 Hello World 程序：Linux Academy”，[在线]。网址：https://dev.ti.com/tirex/explore/node?node=A__Ael48yqnWh8ZfS-MiLdhcA__linux_academy_am62x__XaWts8R__LATEST&search=am62x
11. 德州仪器 (TI)，“入门：AM62xMCU SDK”，[在线]。网址：https://software-dl.ti.com/mcu-plus-sdk/esd/AM62X/latest/exports/docs/api_guide_am62x/GETTING_STARTED.html

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