*Application Note*

# *使用 TXE81xx-Q1 GPIO 扩展器实现 SPI 菊花链连接*

*Tyler Townsend*

**摘要**

本应用手册介绍了使用多个 TXE8124-Q1 器件通过 SPI 菊花链连接的示例用例。

## 内容



## 商标

# 1 简介

菊花链是多种 SPI 协议器件常用的一项功能。菊花链的作用是节省布线成本，同时减少 MCU 和处理器所需的 IO 数量，从而使 PCB 更简洁、设计更简单。

## 2 什么是 SPI 菊花链连接？

SPI 菊花链是一种连接方案，用于与多个串联的 SPI 外设器件进行通信。菊花链连接能够减少所需的导线/布线长度，同时为多个芯片选择节省 MCU 上的 GPIO。

在包含多个外设的普通 SPI 配置中，每个 SPI 外设器件都需要一个芯片选择信号。这意味着系统中的每个 SPI 外设都必须从 MCU 上预留一个 GPIO。请参阅下面示例，了解 4 个外设器件之间的普通 SPI 配置。

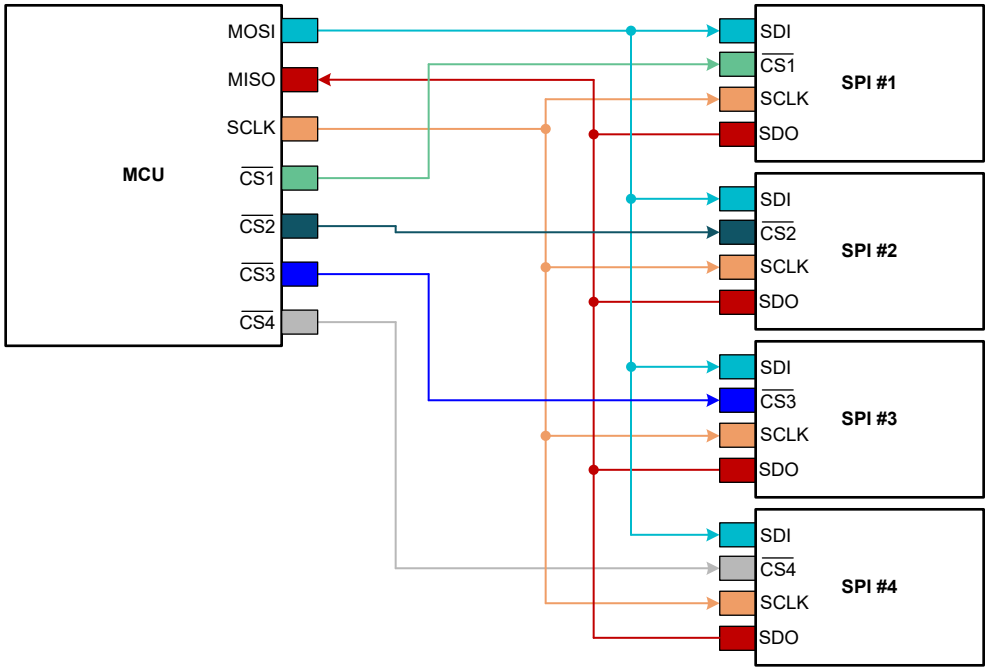


**图 2-1. 不带菊花链的 SPI 总线示例**

控制 4 个 SPI 外设器件需要 MCU 上的 8 个引脚。MCU 的 PICO 引脚连接到所有串行数据输入 (SDI)。MISO 连接到串行数据输出 (SDO)。系统中的所有器件共用时钟引脚 (SCLK)。系统中每个外设都必须有一个独立的芯片选择信号。

SPI 的普通实现需要 MCU 上多个 GPIO 引脚，而这些引脚在某些系统中可能有限。同时，这也意味着系统需要更多的接线将每个片选信号路由到各外设器件。这可能导致物理接线增加，从而增加系统重量，或者使 PCB 布线更杂乱。

为了解决布线过多和芯片选择线路过多的两个问题，可以采用 SPI 菊花链连接方案。IC 器件制造商必须支持 SPI 菊花链功能，该方案才能正常工作。

（注意：SDO 连接到 SDI，只使用一个芯片选择信号。）

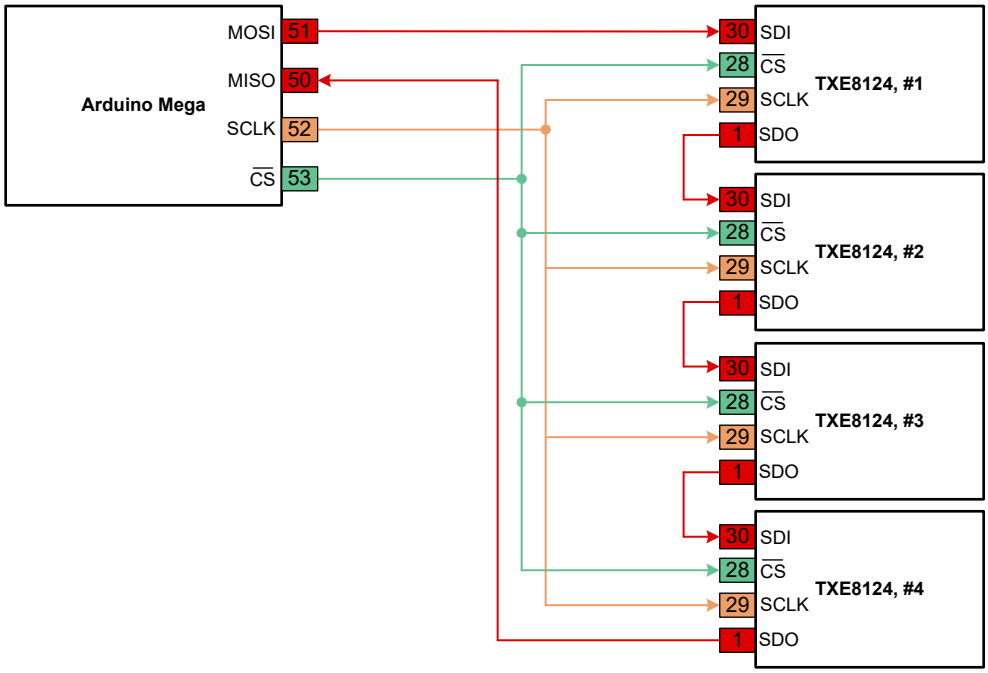


**图 2-2. 使用菊花链的 SPI 示例**

菊花链实现通过将一个外设的输出连接到下一个外设的输入来简化串行数据传输。在这种情况下，由于数据路由不必直接来自 MCU，而是可以“链式”传递到序列中的每个外设，因此布线可以大幅减少。

**备注**

在以菊花链方式连接器件时，对末端器件的 SDO 进行采样可能需要降低 SCLK 频率，因为链中每个器件的数据有效时间会累加，同时还需考虑 SCLK 到达最远端器件的传播延迟。

## 3 使用 TXE81xxEVM 的 SPI 菊花链示例

以下示例演示如何通过将 SDO 和 SDI 线路串联，将 4 块 TXE81XXEVM 连接成菊花链。SCLK 引脚在所有器件之间共享，包括 MCU，同时也共享芯片选择信号。

（电路板从右到左标记为 1-4）

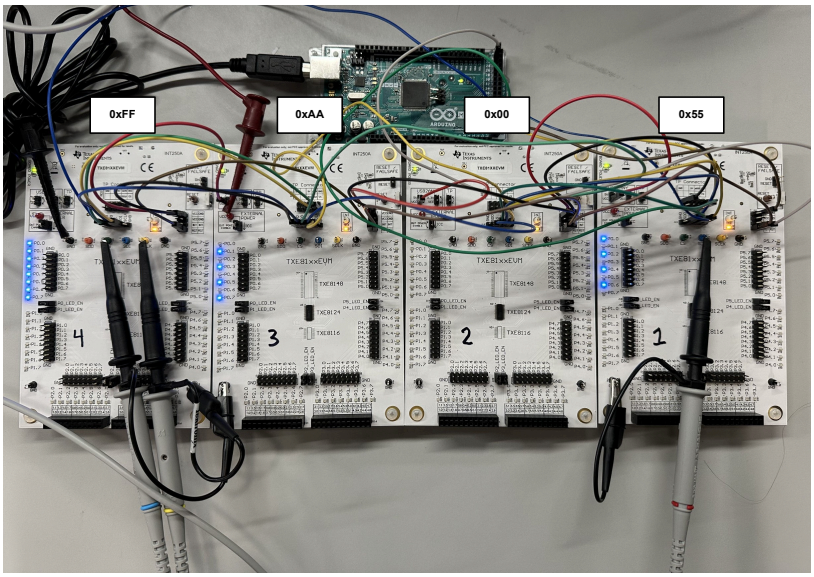


**图 3-1. 4 块 TXE81XXEVM 通过菊花链配置连接**

数据会写入每块 TXE8124-Q1 器件的方向配置寄存器 (0x04)。仅写入端口 0。在方向配置寄存器中设置为 *1* 表示将 GPIO 配置为 OUTPUT。在方向配置寄存器中设置为 *0* 表示将 GPIO 配置为 INPUT。有关菊花链示例中写入的具体数据，请参见表 3-1。

**表 3-1. 板载配置**

| 板型号 | 寄存器地址 | 端口 | 数据 | 输入/输出？ |
|---|---|---|---|---|
| 1 | 0x04 | 0 | 0x55 | 输入 = P0.1、P0.3、P0.5、P0.7 输出 = P0.0、P0.2、P0.4、P0.6 |
| 2 | 0x04 | 0 | 0x00 | 输入 = P0.0 - P0.7 输出 = 无 |
| 3 | 0x04 | 0 | 0xAA | 输入 = P0.0、P0.2、P0.4、P0.6 输出 = P0.1、P0.3、P0.5、P0.7 |
| 4 | 0x04 | 0 | 0xFF | 输入 = 无 输出 = P0.0 - P0.7 |

在 TXE81xx 中，有 4 种 SPI 段类型：状态、标头、地址和数据。表 3-2 说明了在菊花链中发送的每个段的逐位说明。

**备注**

标头分段中的位 15 和 14 是标头 ID，器件控制器使用此 ID 来检测是否正在接收标头分段。标头 ID 位的值为 01，用于指示这是标头分段。位 13 至 5 被保留，位 4 至 0 表示链中的器件数量。可进行菊花链连接的器件数量上限为 31 个。

表 3-2. SPI 段说明

| SPI 段类型 | 位分配 |
|---|---|
| 状态 | 位 [15:14] = 1，表示状态段<br>位 [13:8] = 故障状态寄存器 (0x1900) 的位 5 至 0<br>位 [7:0] = 0，默认 |
| 接头 | 位 [15:14] = 分别为 0 和 1，表示标头段<br>位 [13] = 保留<br>位 [12:0] = 用于确定菊花链中的器件数量 |
| 地址（寄存器地址） | 位 [15] = 表示 SPI 运行模式（1 = 读，0 = 写）<br>位 [14:13] = 不关心 (X)<br>位 [12:8] = 功能地址<br>位 [7] = 不关心 (X)<br>位 [6:4] = 端口选择<br>位 [3:1] = 不关心 (X)<br>位 [0] = 多端口 |
| 数据 | 位 [7:0] = 写入寄存器的数据 |

若要开始通过菊花链发送数据，首先发送标头段，然后发送链中最远电路板的寄存器地址。如果链中有四个器件，则先发送第 4 个器件的寄存器地址，然后发送第 3 个器件的寄存器地址，依此类推。寄存器地址字节发送完成后，接着发送数据字节。第一个数据字节对应链中最远的器件。如果链中有四个器件，第一个数据字节应用于第 4 个器件，第二个数据字节应用于第 3 个器件，依此类推。有关 SPI 数据逐字节发送方式的更详细示例，请参阅图 3-3。

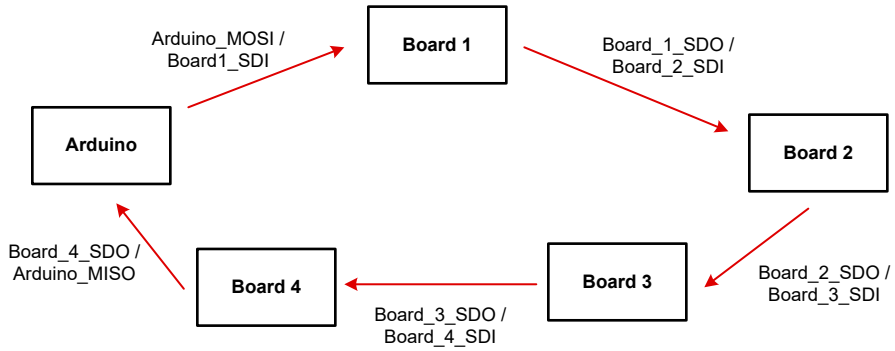


图 3-2. 菊花链方框图

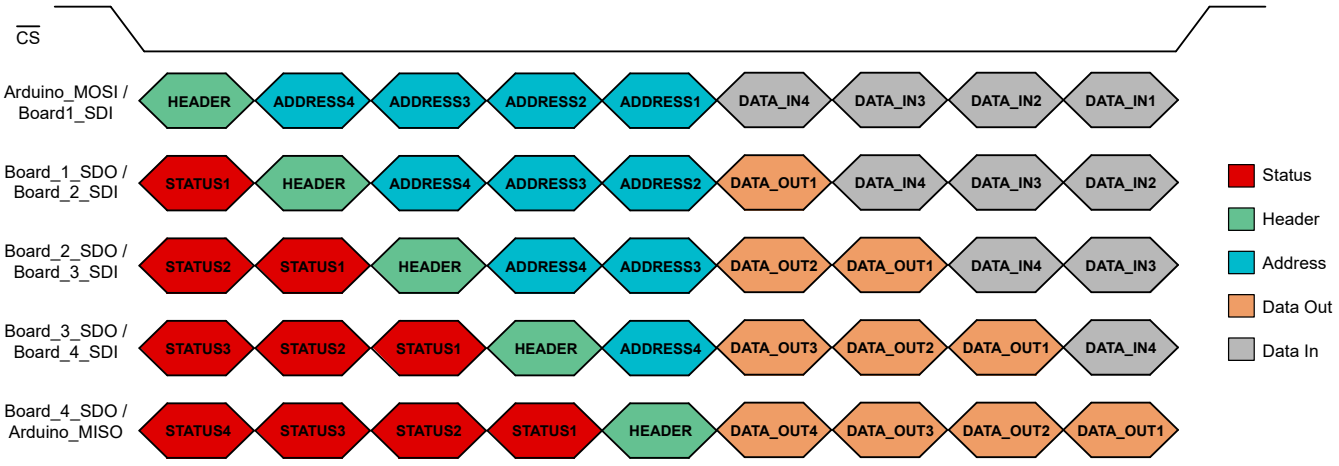


图 3-3. 链中每个字节的序列

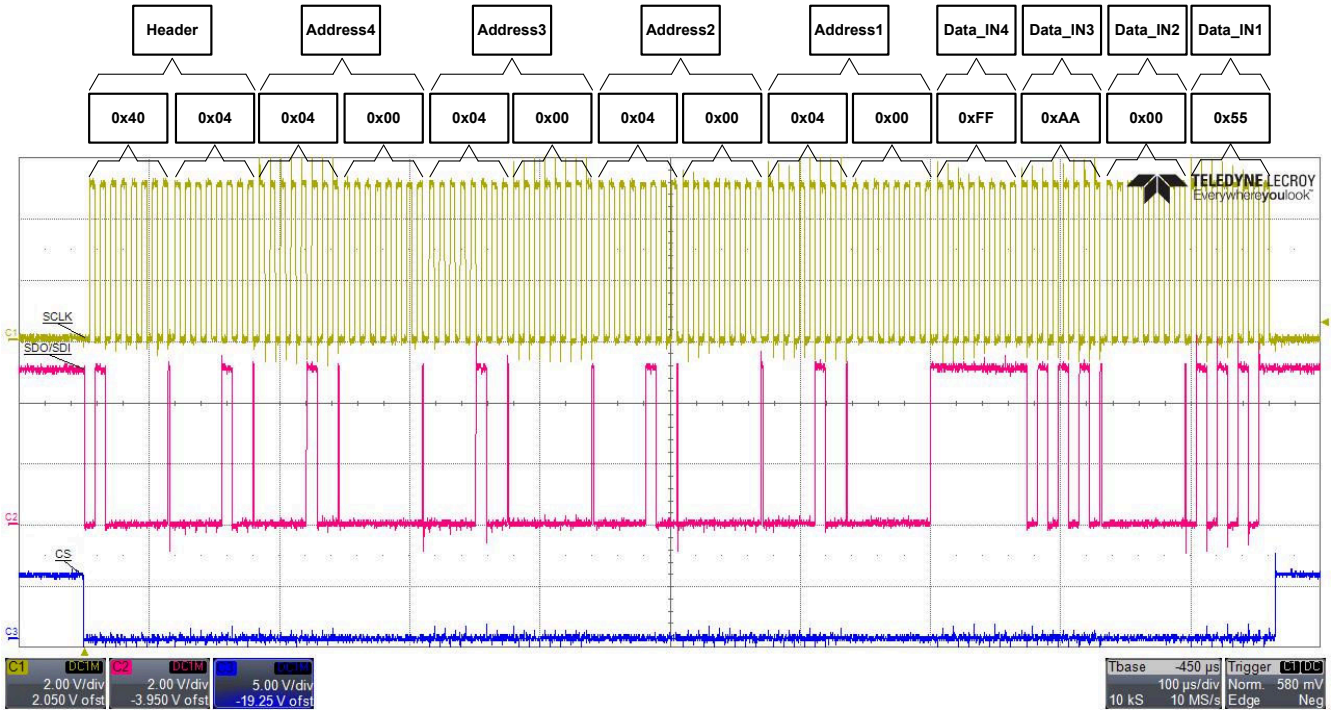


**图 3-4. Arduino_MOSI 到板 1 SDI 的菊花链 SPI 传输带标注波形**

# 4 MSPM0 伪代码示例

```
#include"ti_msp_dl_config.h"//MSP driver library
int main(void)
{
SYSCFG_DL_INT(); //initialize SPI driver
DL_GPIO_setPins(GPIO_LEDS_PORT, GPIO_LEDS_USER_LED_1_PIN | GPIO_LEDS_USER_TEST_PIN); //set /CS HIGH
DL_GPIO_clearPins(GPIO_LEDS_PORT, GPIO_LEDS_USER_LED_1_PIN | GPIO_LEDS_USER_TEST_PIN); //set /CS
LOW
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0b01000000); //header segment
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0b00000100);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 4 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 3 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 2 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x04); //board 1 address
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00);
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0xFF); //board 4 data
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0xAA); //board 3 data
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x00); //board 2 data
DL_SPI_transmitDataBlocking8(SPI_0_INST, 0x55); //board 1 data
DL_GPIO_setPins(GPIO_LEDS_PORT, GPIO_LEDS_USER_LED_1_PIN | GPIO_LEDS_USER_TEST_PIN); //set /CS HIGH
}
```

# 5 Arduino 伪代码示例

## Arduino 编码示例

```
#include <SPI.h>
#define CS 53

//MISO = 50
//CS = 53
//MOSI = 51
//SCLK = 52

void setup() {
  Serial.begin(115200);
  pinMode(CS, OUTPUT);

  SPI.begin();
  SPI.beginTransaction(SPISettings(125000, MSBFIRST, SPI_MODE0));
}

void loop() {
  //send SPI in 8-bit words
  uint8_t header_seg1 = 0b01000000;   //default header segment
  uint8_t header_seg2 = 0b00000100;   //4 devices in the chain
  uint8_t address_seg1 = 0b00000100;  //direction configuration register
  uint8_t address_seg2 = 0b00000000;  //port 0 selected, no multi-port

  digitalWrite(CS, LOW);
  SPI.transfer(header_seg1);
  SPI.transfer(header_seg2);
  SPI.transfer(address_seg1);         //board 4 address
  SPI.transfer(address_seg2);
  SPI.transfer(address_seg1);         //board 3 address
  SPI.transfer(address_seg2);
  SPI.transfer(address_seg1);         //board 2 address
  SPI.transfer(address_seg2);
  SPI.transfer(address_seg1);         //board 1 address
  SPI.transfer(address_seg2);
  SPI.transfer(0xFF);                 //board 4 data
  SPI.transfer(0xAA);                 //board 3 data
  SPI.transfer(0x00);                 //board 2 data
  SPI.transfer(0x55);                 //board 1 data
  digitalWrite(CS, HIGH);
}
```

# 6 总结

与 SPI 菊花链相比，实施常规 SPI 菊花链有几项优缺点，、如下表所示。

**表 6-1. 常规 SPI 与 SPI 菊花链的工程权衡**

| | 常规 SPI | SPI 菊花链 |
|---|---|---|
| 接线 | • 由于存在多条 CS 线路，布线增加<br>• 增加了 PCB 复杂性和潜在布线成本<br>• 导线越多 = 重量越重 = 成本越高 | • 单一 CS 线路可减少布线<br>• 降低 PCB 路由复杂度及与每个 SPI 外设的连接复杂性 |
| 器件控制 | • 可以单独控制每个器件 | • 难以控制单个器件，必须与整个链通信 |
| 数据传输 | • 数据传输本身更快 | • 数据传输本身较慢，因为数据必须通过链中的每个器件 |
| 未来的设计 | • 更改未来设计更困难，每在系统中增加一个外设都需要额外 CS 线路 | • 更容易在 SPI 菊花链末端添加更多外设。仍使用相同的 CS 线路。 |
| 信号完整性 | • 必须将电气连接分散到多个器件，这会导致布线长度/布线距离增加 | • 串联连接可缩短布线长度，从而降低 SI。链中的每个外设器件会将 SPI 数据重新驱动到链中的下一个器件 |
| 软件 | • 软件实现更简单 | • 当链中所有外设都需配置时，可简化数据更新流程<br>• 软件实现更复杂，需要处理更长的 SPI 数据字 |
| 调试 | • 更容易定位外设故障 | • 更难判断链中具体哪个外设损坏<br>• 若链中一个外设故障，可能会影响多个外设 |

# 7 参考资料

-

# 8 修订历史记录

| Changes from Revision * (October 2025) to Revision A (May 2026) | Page |
|---|---|
| • 更新了整个文档中的表格、图和交叉参考的编号格式 | 2 |
| • 向节 2 添加了注释 | 3 |
| • 向节 3 添加了注释 | 5 |

# 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