Application Note

用于 MSPM33C 的 LVGL

摘要

轻量级多功能图形库 (LVGL) 是一种开源图形库，已集成到 MSP M33 软件开发套件 (SDK) 中。此软件解决方案创建了一个独立于硬件并且高度可定制的中间件层。借助 LVGL 和 MSPM33，用户可以轻松创建嵌入式图形用户界面 (GUI)，这些界面可以轻松编写并移植到 MSPM33 微控制器和配套显示屏的任意组合。本应用手册介绍了几个编程示例，以说明如何配置 LVGL 来为多个终端器件 (EE) 创建 GUI。目标是帮助用户了解如何使用 LVGL 库进行 GUI 开发。

代码示例在 MSPM33C321A 上进行了测试，但是，这些示例可以很容易地在任何 MSPM33 器件上运行。大多数示例使用串行外设接口 (SPI) 或四线串行外设接口 (QSPI) 模块将 LVGL 像素数据发送到显示屏，并使用内部集成循环 (I2C)、SPI 或 QSPI 模块从显示屏读取触摸输入。要满足这一要求，可以选择具有支持 SPI 或 QSPI 的集成显示驱动程序的显示屏，以及支持 I2C、SPI 或 QSPI 的集成触控驱动器。内置足够数字线路来解码 SPI/QSPI 信号的内窥镜对于调试非常重要。

内容

插图清单

商标

1 概述

LVGL 提供了针对嵌入式系统优化的轻量级开源图形库。凭借高性能渲染能力、灵活的 UI 组件及广泛的硬件兼容性，LVGL 为资源受限器件实现流畅、现代化的 GUI 开发提供支持。德州仪器 (TI) 的客户借助 LVGL 的高效内存管理、硬件加速支持及易集成特性，能以最少的开发投入，打造出响应迅速且视觉效果出众的界面。

LVGL 已作为可选中间件库集成至 MSP M33 SDK 中。要使用 LVGL，必须考虑以下主要因素：

- 工程设置
- 配置
- 初始化
- 图形输出

- 图形输入
- 图形更新

此架构在图 1-1 中进行了介绍。

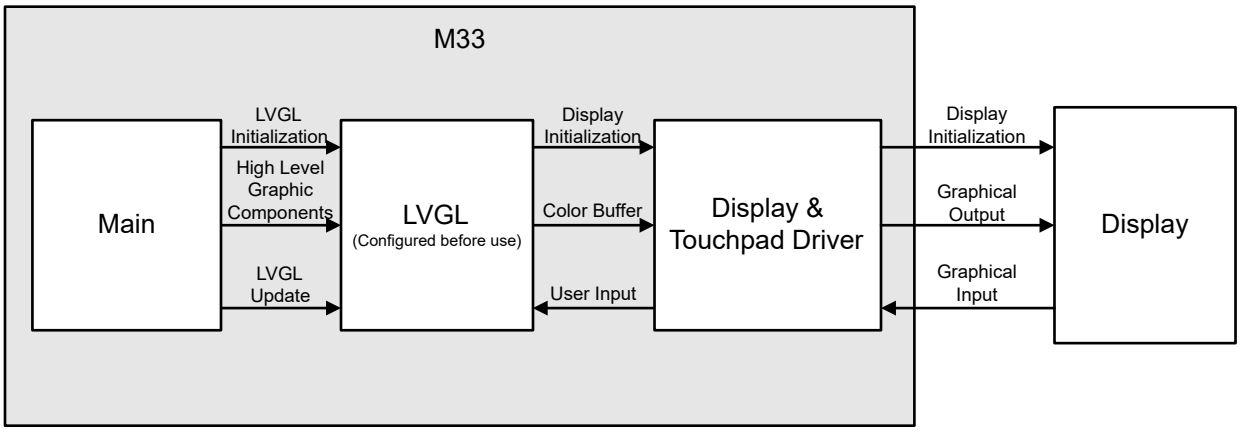

图 1-1. M33 LVGL 架构

1.1 LVGL 项目设置

LVGL 已集成到 M33 SDK 中，但是必须将库链接到工程中。若要执行此链接，可以按照以下步骤操作：

- 右键点击工程，然后选择“Add Files/Folders...”
- 在“Add Files/Folders”窗口中，选择“+”符号以添加 LVGL 库
- 从“Select files/folders”窗口中，将下拉列表从“Select files to add”更改为“Select folders to link”
- 然后单击“Path”附近的“…”图标，选择 SDK LVGL 库的基本路径。
  - SDK LVGL 库位于“C:\ti\mspm33_sdk_XX\source\third_party\lvgl”，其中“XX”是最新的 SDK 版本。
- 单击“Select files/folders”窗口上的“Ok”
- 最后，在“Add Files/Folders”窗口中单击“Ok”
- SDK LVGL 库目前应该已包含在工程中

该库具有最基本的配置，没有低级驱动程序。要针对特定应用配置库并为特定显示添加低级驱动程序，必须在工程中添加五个文件并进行调整：

- lv_conf.h：LVGL 的主配置文件
- lv_port_disp.h：LVGL 显示驱动程序的头文件
- lv_port_disp.c：LVGL 显示驱动程序的源文件
- lv_port_indev.h：LVGL 输入器件驱动程序的头文件
- lv_port_indev.c：LVGL 输入器件驱动程序的源文件

1.2 配置

LVGL 具有高度可配置性，欺诈库的配置在 *lv_conf.h* 文件中完成。此文件中有多个字段必须更新，才能使库正常工作。这些字段包括：

- MY_DISP_HOR_RES：用于根据所用显示屏设置内部颜色缓冲区的水平尺寸
- MY_DISP_VER_RES：用于根据所用显示屏设置内部颜色缓冲区的垂直尺寸
- LV_COLOR_DEPTH：用于根据所用显示屏设置颜色缓冲区的颜色格式
- LV_MEM_SIZE：用于设置用来存储对象和动画的动态内存量，通常为 48kB
- LV_DRAW_COMPLEX：用于启用复杂形状，如圆角、圆和圆弧

此外，还可以调整几个 *lv_conf.h* 部分以减少应用程序内存占用量。这些字段包括：

- 字体：禁用未使用的字体以减少所需闪存大小
- 小部件：禁用未使用的小工具/演示以减少所需闪存大小
- 演示：禁用未使用的演示以减少所需闪存大小

1.3 初始化

使用 LVGL 创建嵌入式 GUI 时，必须初始化正在使用的显示，并且必须将 LVGL 配置为与显示配合使用。这种初始化和配置通过 LVGL 全局函数 *lv_port_disp_init* 完成。此函数分为三个部分：

- 显示初始化
  - LVGL 包含一个全局函数 *disp_init*。此函数在 *lv_port_disp_init* 中调用，可用于触发所使用显示屏的初始化序列。
- 颜色缓冲区分配和刷新模式选择
  - LVGL 使用颜色缓冲区来存储渲染图像的像素数据。该缓冲区 lv_disp_draw_buf_t 在 *lv_port_disp_init* 中分配。有两种典型的缓冲配置：
    - 单路缓冲区：一个缓冲区被分配并传递给 *lv_disp_draw_buf_init*。该缓冲区的大小通常是水平结果的十倍（10 行像素）。该缓冲区仅填充渲染图像的像素数据，CPU 负责将颜色缓冲区的像素信息传输到用于与显示屏通信的通信接口。
    - 具有 DMA 的双缓冲区：两个缓冲区分配了相同大小的内存，并传递到 *lv_disp_draw_buf_init*。两个缓冲区的大小通常为 10 行像素。LVGL 使用重建图像的部分像素数据填充一个缓冲区，然后使 DMA 可以使用这些数据开始将数据传输到通信接口。
  - 还有两种模式可以用于配置 LVGL 执行图像刷新的方式：
    - 部分刷新：在部分刷新中，只有正在更新的渲染图像部分被写入颜色缓冲区。默认情况下启用此项。该模式支持更快的刷新和更小的颜色缓冲区，然而，如果显示部分撕裂或条纹，则可能会导致残留的显示伪影。
    - 完全刷新：在完全刷新时，整个渲染的图像将被写入颜色缓冲区。这可通过设置 *lv_disp_drv_t* 显示驱动程序中的 *full_refresh* 字段来启用。此模式可消除残留显示伪影的问题，但速度要慢得多，并且需要一个与屏幕分辨率相等的颜色缓冲区。
- 显示驱动程序配置
  - LVGL 提供了必须配置的显示驱动程序结构，以便 LVGL 正确呈现显示。第一步是通过使用驱动程序的新实例 *lv_disp_drv_t* 调用 *lv_disp_drv_init*，来初始化显示驱动程序
  - 初始化显示驱动程序后，必须配置的主要值包括：
    - 水平和垂直分辨率：水平和垂直分辨率 *hor_res* 和 *ver_res* 是显示驱动程序的成员，可以使用之前定义的 MY_DISP_HOR_RES and MY_DISP_VER_RES 进行设置
    - 刷新回调：必须使用刷新回调 *flush_cb* 的位置来配置显示驱动程序。刷新回调是在 LVGL 刷新已填充渲染图像像素数据的颜色缓冲区时调用的方法。此回调方法将在本应用手册中进一步讨论
    - 绘制缓冲区：显示驱动程序必须配置先前初始化的颜色缓冲区即 *draw_buf* 的位置
    - 刷新模式：如果使用完全刷新模式，则必须将显示驱动程序配置为 *full_refresh* 设置为 1
  - 最后，必须通过调用 *lv_disp_drv_register* 向 LVGL 注册显示驱动程序

LVGL 支持多种类型的输入器件，包括触控板、鼠标、键盘、编码器、或按钮。所有这些输入器件都必须初始化并配置 LVGL。这是在 LVGL 全局函数 *lv_port_indev_init* 中执行的，它分为两部分：

- 输入器件初始化
  - LVGL 包含几个可用于初始化输入器件的全局函数。例如，当使用具有电阻式或电容式触控的显示屏时，可以调用全局函数 *touchpad_init* 来执行触控板初始化。
- 输入器件驱动程序配置
  - LVGL 提供了必须配置的输入器件驱动程序结构，以便 LVGL 正确处理输入。第一步是初始化输入器件驱动程序，方法是使用驱动程序的新实例 *lv_indev_drv_t* 调用 *lv_indev_drv_init*
  - 初始化输入器件驱动程序后，必须配置的主要值为：
    - 输入器件类型：所使用的输入器件类型：指针（触控板或鼠标）、键盘、按钮和编码器
    - 输入读回调：输入器件驱动程序必须配置输入读回调 *read_cb* 的位置。输入读取回调是 LVGL 为输入器件的任何输入进行读取时调用的方法。此回调方法将在本应用手册中进一步讨论
  - 最后，必须通过调用 *lv_indev_drv_register*，使用 LVGL 注册输入器件驱动程序

1.4 LVGL 输出

当 LVGL 准备好刷新颜色缓冲区时，LVGL 将调用在配置步骤中注册的刷新回调方法。刷新回调用于处理颜色缓冲区并将其发送至特定于显示屏的通信接口。典型的处理方法包括：

- 单个像素刷新：像素数据逐个写入通信接口，并发送到显示屏。这种方法最简单，但也最慢，因为 CPU 需要处理时间
- 全缓冲区刷新：像素数据通过 DMA 传输到通信接口并发送至显示屏。由于采用 DMA，这种方法比前一种方法更复杂，但是，由于 DMA 的使用，LVGL 可以开始将渲染图像的下一部分写入第二个颜色缓冲区，这使得这种方法更快。

处理颜色缓冲区后，必须通知 LVGL 可以通过调用 *lv_disp_flush_ready* 来刷新颜色缓冲区。

1.5 LVGL 输入

当 LVGL 准备好从输入器件读取输入时，LVGL 将调用在配置步骤中注册的输入读取回调方法。输入读取回调用于轮询输入器件以便获取输入信息。从输入器件读取数据的典型流程如下：

1. 检查输入器件是否已进行交互，并通过相应的交互更新输入器件驱动程序状态
2. 从输入器件检索输入信息。这通常是触控板的 x/y 位置或键盘的键编号。使用检索到的输入信息，设置与输入器件类型相对应的输入器件驱动程序字段

1.6 LVGL 更新

为了使 LVGL 能够正确执行动画和其他任务，LVGL 需要系统节拍、定期计时器触发器和睡眠管理。下面列出了有关这些要求的其他信息：

- 系统节拍：LVGL 要求通知它已经过的时间。这是通过定期调用 lv_tick_inc 函数来完成的，经过的时间为毫秒。此操作可以在计时器中断或主 while 循环中完成。
- 周期性计时器触发：LVGL 要求它有专门的 CPU 时间来处理计时器及任务。这是通过以类似于系统节拍的方式定期调用 lv_timer_handler 来实现的。
- 睡眠管理：在某些情况下，LVGL 要求在 MCU 进入睡眠模式时通知它。如果使用计时器中断调用 lv_tick_inc 或 lv_timer_handler，则必须停止此中断，以便在 MCU 唤醒时 LVGL 返回到相同状态。

2 LVGL 示例

以下 LVGL 示例展示了连接到 TFT 显示屏的 LP-MSPM33C321A，该显示屏采用 ST7796 显示驱动器和 DFT6636U 电容式触控驱动器。ST7796 显示驱动器通过 SPI 与 LP-MSPM33C321A 进行通信，DFT6636U 触控驱动器通过 I2C 接口与 LP-MSPM33C321A 进行通信。

2.1 硬件连接

LP-MSPM33C321A 和 TFT 显示屏之间需要进行的连接记录在表 2-1 中。

表 2-1. LVGL 硬件连接示例

LP-MSPM33C321A	TFT 显示
PA8	SPI SCLK
PA9	SPI PICO
PA10	复位
PB3	PWM
PB6	LCD SCS
PB30	LCD SDC
PC5 (未使用)	INT
PA0	SDA
PA1	SCL
PC4	RST

2.2 软件

下面描述了对 LVGL 示例所做的软件配置：

- 配置
  - MCU：以下值是专门针对该示例配置的：
    - GPIO：本示例中有几个 GPIO 引脚配置为初始设置为高电平或低电平，如表 2-2 中所定义。这些值取决于显示屏及触控驱动器的要求。

表 2-2. LVGL 示例 GPIO 引脚配置

引脚名称	方向	初始值	分配的引脚
LCD_SCS	输出	设置	PB6
LCD_SDC	输出	被清零	PB30
LCD_RESET	输出	设置	PA10
LCD_PWM	输出	被清零	PB3
CTP_RST	输出	设置	PC4
CTP_INT	输入	不适用	PC5

    - SYSCTL：SYSCTL 模块已配置成使用外部高频时钟输入作为 MCLK 的输入。HFXT 设置为以 40MHz 运行
    - I2C：所选的 I2C 外设 UC3 已配置为作为控制器运行并在快速模式 (400kHz) 下运行，欺诈 SDA 和 SCL 线路分别分配给 PA0 和 PA1
    - SPI：所选的 SPI 外设 UC2 已配置为用作控制器，在 40MHz 下运行，使用 CS1 作为芯片选择，并在 Motorola 4 线模式下运行，欺诈 SCLK、PICO、POCI 和 CS1 分配给 PA8、PA9、PB19 和 PB6
    - 计时器：所选的计时器模块 TIMG8_0 已配置为以 BUSCLK 为源，预分频为 40 倍，并以 1ms 的周期进行定期向下计数。中断已被配置成在计时器发生零事件时触发

- LVGL：
  - 在此示例中，LVGL 已配置为所使用的 TFT 显示屏的规格。由于 LVGL 演示使用了许多不同的模块，因此许多可选字体和小工具都已启用。演示配置值 LV_USE_DEMO_WIDGETS 也经过专门启用，以便编译和运行 LVGL 演示。

```
#define MY_DISP_HOR_RES         480
#define MY_DISP_VER_RES         320
...
#define LV_COLOR_DEPTH           16
...
#define LV_MEM_SIZE   (48U * 1024U) // [bytes]
...
#define LV_DRAW_COMPLEX 1
...
#define LV_USE_DEMO_WIDGETS 1
```

- 初始化
  - 需要先初始化的外设是之前通过 SysConfig 配置的外设
  - 接下来，用于触发系统节拍的计时器中断需要初始化
  - 在外设和计时器之后，需要初始化 LVGL 特定项
    - LVGL 初始化：LVGL 需要通过调用 *lv_init* 来进行初始化
    - 输出初始化：通过调用 *lv_port_disp_init* 来初始化 LVGL 显示驱动程序
      - 在本例中，*lv_port_disp_init* 遵循在节 1.3 中定义的初始化步骤。刷新回调是在本节后面定义的

```
void lv_port_disp_init(void)
{
    disp_init(); // Initialize your display

    static lv_disp_draw_buf_t draw_buf_dsc_1;
    static lv_color_t buf_1[MY_DISP_HOR_RES * 10];                          /*A buffer
for 10 rows*/
    lv_disp_draw_buf_init(&draw_buf_dsc_1, buf_1, NULL, MY_DISP_HOR_RES * 10);   /
*Initialize the display buffer*/

    static lv_disp_drv_t disp_drv;                          /*Descriptor of a display
driver*/
    lv_disp_drv_init(&disp_drv);                    /*Basic initialization*/

    disp_drv.hor_res = MY_DISP_HOR_RES;
    disp_drv.ver_res = MY_DISP_VER_RES;

    disp_drv.flush_cb = disp_flush;

    disp_drv.draw_buf = &draw_buf_dsc_1;

    lv_disp_drv_register(&disp_drv);
}
```

    - 输入初始化：通过调用 *lv_port_indev_init* 来初始化 LVGL 输入器件驱动程序
      - 在本例中，*lv_port_indev_init* 遵循节 1.3 中定义的初始化步骤。本节稍后会定义输入读取回调

```
void lv_port_indev_init(void)
{
    static lv_indev_drv_t indev_drv;

    touchpad_init(); //Initialize your touchpad

    lv_indev_drv_init(&indev_drv); //Register a touchpad input device
    indev_drv.type = LV_INDEV_TYPE_POINTER;
    indev_drv.read_cb = touchpad_read;
    indev_touchpad = lv_indev_drv_register(&indev_drv);
}
```

  - 最后，系统节拍计时器启动

```
int main(void)
{
```

```
    SYSCFG_DL_init();

    NVIC_EnableIRQ(TIMER_0_INST_INT_IRQN);

    lv_init(); // init LVGL
    lv_port_disp_init();
    lv_port_indev_init();

    DL_TimerG_startCounter(TIMER_0_INST);
```

- 输出
  - LVGL 通过调用先前在初始化部分中定义的刷新回调方法来发送像素数据。在此示例中，*disp_flush* 调用低级驱动程序代码 *lvgl_LCD_Color_Fill*，它处理 SPI 通信，然后通知 LVGL，刷新已准备就绪

```
static void disp_flush(lv_disp_drv_t * disp_drv, const lv_area_t * area, lv_color_t * color_p)
{
    lvgl_LCD_Color_Fill(area->x1, area->y1, area->x2, area->y2, color_p);
    lv_disp_flush_ready(disp_drv); //Inform the graphics library that you are ready with the
flushing
}
```

- 输入
  - LVGL 通过调用先前在初始化部分中定义的输入读取回调方法来尝试读取用户输入。在此示例中，*touchpad_read* 调用处理 I2C 通信的低级驱动程序代码，以检查触控板是否已按下，然后在发生按下时处理按下的位置

```
static void touchpad_read(lv_indev_drv_t * indev_drv, lv_indev_data_t * data)
{
    static lv_coord_t last_x = 0;
    static lv_coord_t last_y = 0;

    if(touchpad_is_pressed()) {
        touchpad_get_xy(&last_x, &last_y); //Save the pressed coordinates and the state
        data->state = LV_INDEV_STATE_PR;
    }
    else {
        data->state = LV_INDEV_STATE_REL;
    }

    data->point.x = last_x; //Set the last pressed coordinates
    data->point.y = last_y;
}
```

- 更新
  - LVGL 系统节拍是通过先前配置的 TIMG8_0 计时器中断而触发的。此中断每毫秒触发一次。

```
void TIMER_0_INST_IRQHandler(void)
{
    switch (DL_TimerG_getPendingInterrupt(TIMER_0_INST)) {
        case DL_TIMER_IIDX_ZERO:

            lv_tick_inc(1);     //LVGL Heart Beat

            break;
        default:
            break;
    }
}
```

  - LVGL 计时器触发在示例的主 while 循环中处理。为了不影响所有 CPU 处理，会有 320 个周期的延迟。

```
while (1) {
        // update LVGL task
        lv_task_handler();

        delay_cycles(320);
    }
```

- 主演示

- 主演示在 *lv_demo_widgets* 中定义，它包含在 LVGL 库中。此演示展示了 LVGL 中包含的主要小部件。为了使用此演示，本演示的主源文件中包含了演示的头文件。

```
#include "lv_demo_widgets.h"
...
int main(void)
{
...
    lv_demo_widgets();
...
}
```

2.3 LVGL 示例摘要

在 LP-MSPM33C321A 和 TFT 显示屏之间进行硬件连接并且在 M33 器件上编译并刷写软件更改后，M33 和 TFT 显示屏相互通信以运行 LVGL 演示。在此演示中，可以交互显示多个小工具，包括按钮、图表和图像。

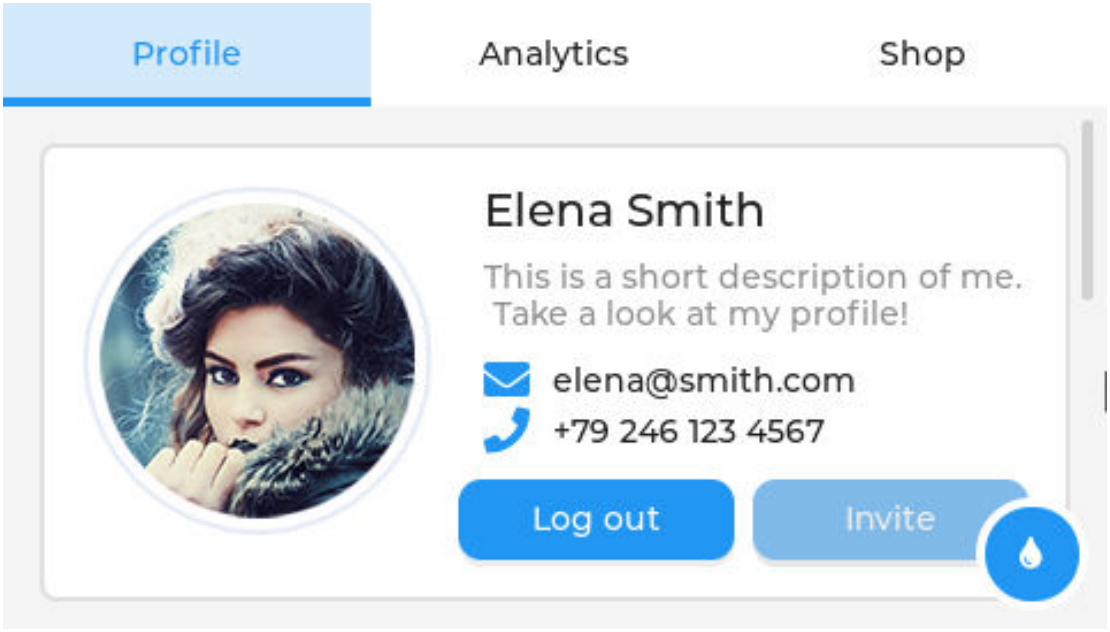

图 2-1. LVGL 演示用户界面

3 总结

本文档首先概述 LVGL 以及正确将其集成至 MSPM33 嵌入式 GUI 所需注意事项。然后，本应用手册介绍了 LVGL 与 SPI 显示驱动程序和 I2C 触控板驱动程序配合使用以复制基本 LVGL 演示的示例。

4 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

日期	修订版本	注释
2025 年 12 月	*	初始发行版

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