User's Guide

TLC6962x/TLC6963x-Q1 示例代码

TEXAS INSTRUMENTS

摘要

本文档介绍了准备和使用与 LAUNCHXL-F280039C 配对使用时的 TLC6962x/TLC6963x-Q1 器件系列示例代码。按照提供的设置说明，安装的代码会点亮 EVM 上的 LED。TLC696[2|3][3|6|9] 器件可与 TLC69698-Q1 配对使用，以使用 MCU 的标准 SPI 或 UART 模块与 LED 驱动器通信。TLC696[2|3][1|2|4|5|7|8] 器件可与 TLC69697-Q1 配对使用，以使用 MCU 的标准 UART 模块与 LED 驱动器通信，并具备使用物理 CAN 层在多块板之间进行通信的能力。

内容

插图清单

表格清单

商标

所有商标均为其各自所有者的财产。

1 简介

示例代码展示了点亮 TLC69628Q1EVM、TLC69622Q1EVM、TLC69625Q1EVM 和 TLC69698Q1EVM 上的 LED 的功能。每个 EVM 都有专用的示例代码。但是，唯一的区别在于在 `led_driver.h` 文件中选择了所用的 LED 驱动器 IC。这有助于用户无需对示例代码进行任何修改即可点亮 EVM。

代码中有两种模式：动画和简单测试。默认选择动画模式。节 3.3 介绍了如何在这两种模式之间切换。在动画模式下，一个帧用于向左、向右、向上和向下滚动，并根据预定义的顺序淡入淡出。对于 TLC69628Q1EVM、TLC69622Q1EVM 和 TLC69625Q1EVM，该帧是一个 99x6 像素的德州仪器 (TI) 标识。对于 TLC69698Q1EVM，该帧是一个 99x8 像素的德州仪器 (TI) 标识。这意味着 EVM 的 LED 显示屏上只显示整个帧的一部分。有关这方面的示例，请参阅节 3.6。如何在示例代码中生成该帧的方法不在本文档的讨论范围内。

在简单测试模式下，用户可以使用一些预定义的 API 来点亮 LED 板、执行诊断或构建定制的接口命令。示例代码具有使用 50% 占空比点亮所有 LED 的功能。此外，还会执行诊断，以确定任何 LED、LED 驱动器故障，以及在启用时确定通用插入式增强连接 (UPAC) 器件故障。更多有关诊断结果的信息，请参阅节 3.5。预定义的 API 会根据指定的系统自动调整。如需了解有关系统规格的更多详情，请参阅节 3.3 和节 3.4。

TLC69698Q1EVM 将 TLC69698-Q1 UPAC 器件与一个 TLC69629-Q1 LED 驱动器和一个 TLC69623-Q1 LED 驱动器配对使用。TLC69698Q1EVM 的示例代码已启用该配对，并使用 LAUNCHXL-F280039C 的 SPI 与 TLC69698-Q1 进行通信。要更改为 UART，请遵循硬件用户指南中描述的跳线设置。软件更改在节 3.3 中描述。使用 UPAC 器件可启用额外的诊断功能，节 3.5 对此进行了更详细的描述。

2 软件设置

要为 TMS320F280039C LaunchPad™ 设置软件，请执行以下步骤（在装有 Windows 10 操作系统的计算机上演示）：

1. 下载并安装 Code Composer Studio™ (CCS)。
   a. 下载 Code Composer Studio 集成开发环境 (IDE)（版本 12.7.0）。
   b. 按照安装说明安装 Code Composer Studio。在安装过程中，如果将“Setup type”选择为“Custom Installation”，请确保在“Select Components”中选择“C2000 real-time MCUs”，如图 2-1 中的红色框所示。

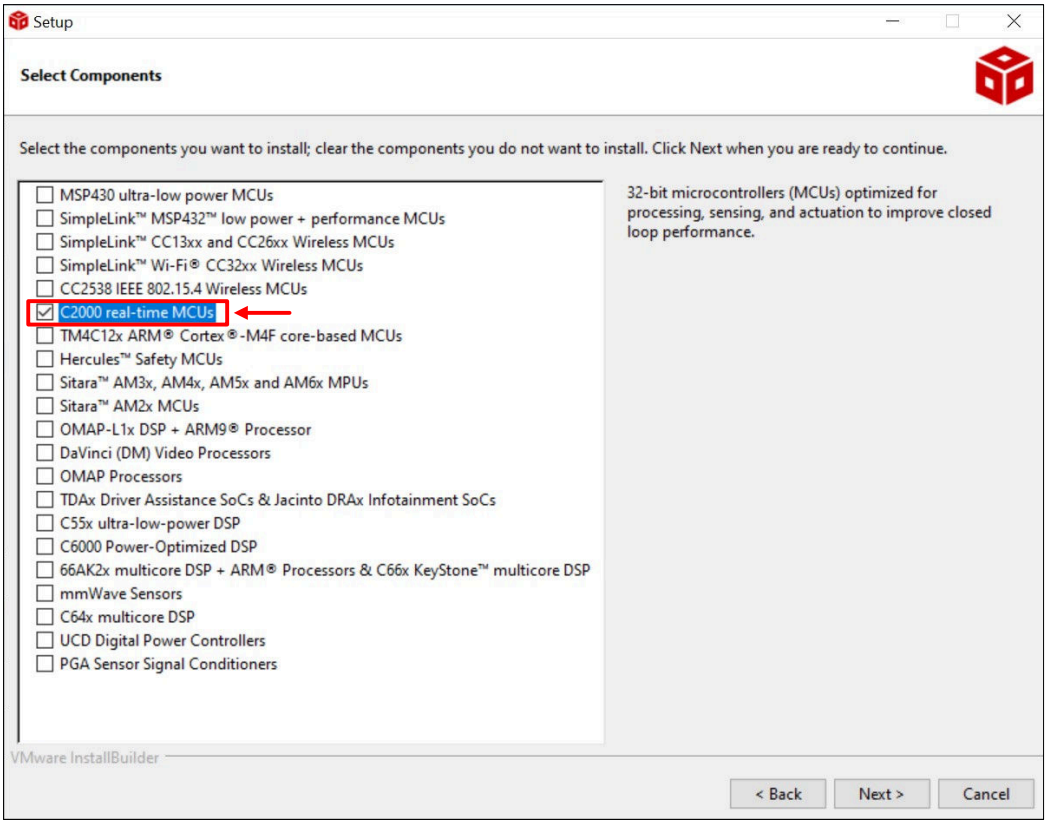

图 2-1. Code Composer Studio 安装过程

2. 下载并安装 C2000Ware（版本 5.02.00.00）。
   a. 要验证安装是否正确，请打开 Code Composer Studio 软件。点击顶部菜单栏中的“Windows”。然后从下拉列表中点击“Preferences”。将显示“Preferences”窗口。从左侧栏中，依次选择“Code Composer Studio”->“Products”。
   b. 确保“Discovered products”下有“C2000Ware 5.2.0.00”和“SysConfig”（安装 C2000Ware 时会自动安装 SysConfig），如图 2-2 所示。如果没有出现“C2000Ware 5.2.0.00”或“SysConfig”，请点击“Preference”窗口右侧的“Refresh”。如果仍然不出现，请检查安装是否正确。SysConfig 的独立桌面版本可在 SYSCONFIG 系统配置工具中找到。

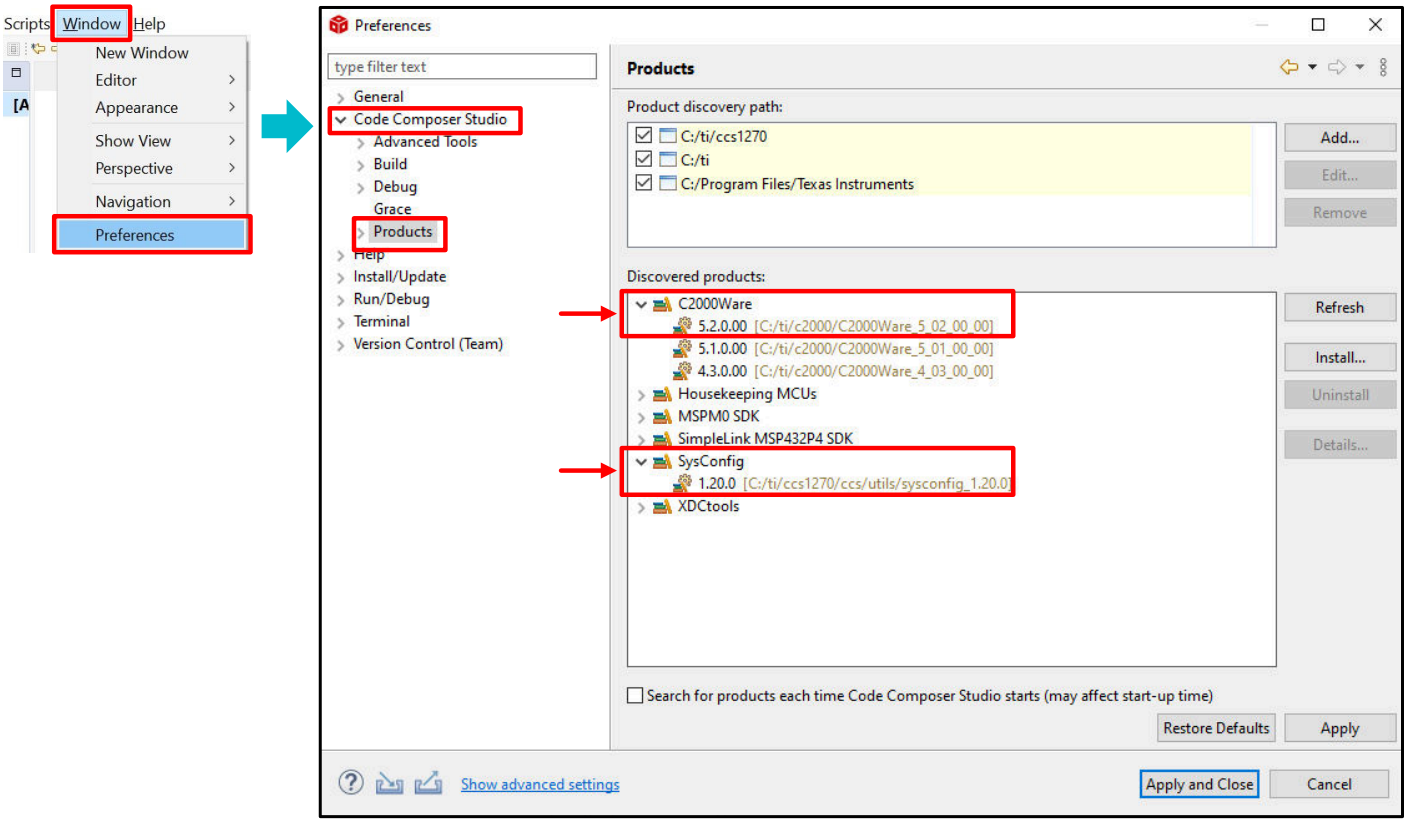

图 2-2. C2000Ware 5.02.0.00 安装验证

3. 下载并导入示例代码。
   a. 根据链接中提供的过程导入 Code Composer Studio (CCS) 工程：导入 CCS 工程。
4. 在加载和运行程序之前，请将 LAUNCHXL-F280039C 上开关 S2 的 SEL1 和 SEL2 都设置为 1 位置。
5. 根据链接中提供的过程加载程序：构建和运行工程。

3 示例代码结构

3.1 设计参数

表 3-1 中列出了用于 EVM 的 LED 矩阵显示设计参数。

表 3-1. EVM 设计参数

设计参数	TLC69628Q1EVM	TLC69698Q1EVM	TLC69622Q1EVM	TLC69625Q1EVM
显示模块大小	16 x 6 颗白光 LED	7 x 8 颗白光 LED	4 x 2 颗白光 LED	4 x 6 颗白光 LED
帧速率	60Hz	60Hz	60Hz	60Hz
PWM 分辨率	16 位	16 位	16 位	16 位
级联器件	2	2	1	1
CLK_I 频率	3MHz	3MHz	3MHz	3MHz

3.2 流程图

图 3-1 描述了示例代码中的高级流程。在设置 MCU 期间，一些输入来自文件 `system_info.h`。根据 *IF_SEL* 宏，MCU 会设置 SPI 模块或 UART 模块。当选择 UART 时，会选择默认数据速率 515625bps。当选择 SPI 时，SPI 通信的时钟频率使用 *SPI_FREQ_IN_HZ* 宏。更多有关 `system_info.h` 文件的详细信息，请参阅节 3.3 和节 3.4。

当要写入 LED 驱动器寄存器时，大多数数据来自 `LED_Reg_settings.h` 文件。但是，级联器件的数量（寄存器 DEVSET 中的字段 CHIP_NUM）来自文件 `system_info.h`。

该流程图还展示了 `frames.c` 和 `frames.h` 文件，其中包含在动画模式期间使用的帧。如何在示例代码中生成该帧的方法不在本文档的讨论范围内。

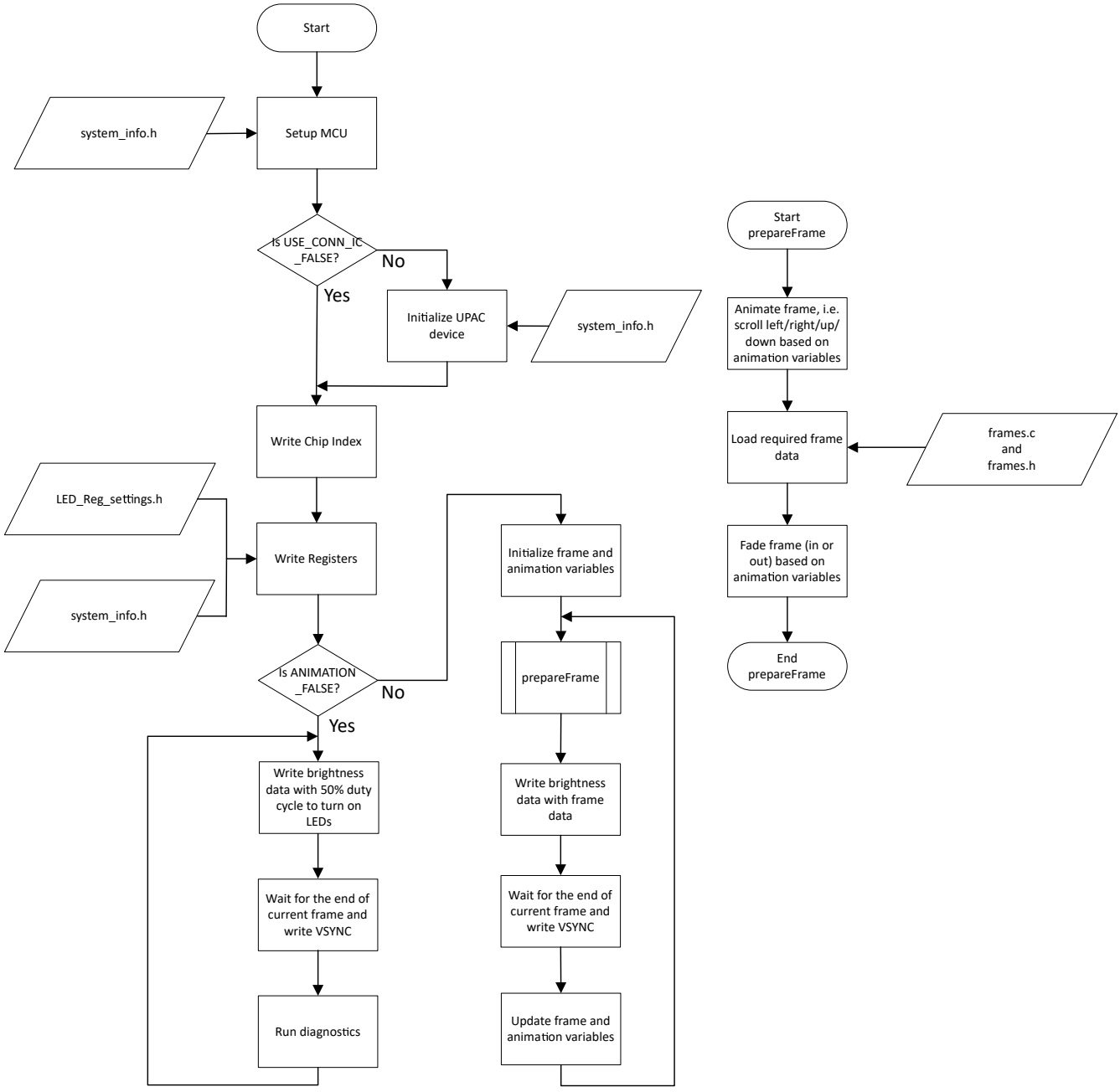

图 3-1. 示例代码流程图

3.3 基本系统设置

本节介绍了示例代码如何设置不同的参数来识别系统的构建方式。

第一部分是实际使用的 LED 驱动器 IC。在 `led_driver.h` 文件中，选择所使用的 LED 驱动器 IC。

```
#include "TLC69628.h"
```

代码支持以下种产品：

- TLC6962[1|2|3|4|5|6|7|8|9]
- TLC6963[1|2|3|4|5|6|7|8|9]

请注意，对于“Q1”器件，不会添加此名称，仅使用基本产品名称。

表 3-2 中列出了影响系统设置和宏及变量位置的宏和变量的摘要。

表 3-2. 每个文件的宏和变量名称的摘要

文件名	宏/变量名称	说明
system_info.h	*SPI_FREQ_IN_HZ*	当 *IF_SEL* 定义为 *SPI* 时，表示 CLK_I 频率（LED 驱动器）或 SCLK_RX 频率 (TLC69698-Q1)。
	USE_CONN_IC	在使用 UPAC 器件 TLC69698-Q1/TLC69697-Q1 时选择。
	RUN_MODE	在不同的代码运行模式之间进行选择。支持的选项包括 *ANIMATION* 和 *SIMPLE_TEST*。
	IF_SEL	选择用于通信的 MCU 接口。
	TOTAL_SCAN_LINES	扫描线数。对于 TLC6962x/TLC6963x-Q1 系列，该值始终为 1。
	BUS_NUM	菊花链总线数量。
	CASCADED_UNITS_BUS1	菊花链总线 1 中的器件数量。
	CASCADED_UNITS_BUS2	菊花链总线 2 中的器件数量。
	CASCADED_UNITS_BUS3	菊花链总线 3 中的器件数量。仅支持带 UPAC 器件的 UART。
	CASCADED_UNITS_BUS4	菊花链总线 4 中的器件数量。仅支持带 UPAC 器件的 UART。
	SCAN_FET_CTRL_NUM	一个菊花链中的 SCAN FET 控制器数量。对于 TLC6962x/TLC6963x-Q1 系列，该值始终为 0。
system_info.c	*FRAME_RATE*	VSYNC 命令的间隔。
	dev_addr	UART 总线上使用的连接器件的器件地址。

在文件 `system_info.c` 中，指定了 UPAC 器件的帧速率和 UART 器件地址。帧周期以 Hz（每秒帧数）为单位。受支持的最小帧速率为 1Hz。

```
uint32_t FRAME_RATE = 60;       // 60Hz frames-per-second
const uint16_t dev_addr[BUS_NUM] = {DEVICE_ADDR__0};
```

只有当 *USE_CONN_IC* 设置为 _TRUE 且 *IF_SEL* 设置为 *UART* 或 *UART_CAN* 时，才需要设置变量 *dev_addr*。该变量指定 UART 总线上 UPAC 器件的地址。

文件 `system_info.h` 包含多个系统定义。

```
// Desired CLK_I or SCLK_RX frequency
// Supported range is 500kHz to 7.5MHz --> For higher frequencies need to enable SPI high speed mode
// Note: Exact frequency is not always possible
#define SPI_FREQ_IN_HZ        3000000
```

宏 *SPI_FREQ_IN_HZ* 定义了 SPI 运行的数据速率，即 LED 驱动器的引脚 CLK_I 或 TLC69698-Q1 的引脚 SCLK_RX 的时钟频率。该频率是 MCU 系统频率的整数分频器。因此，实际 SPI 频率有时不同于 *SPI_FREQ_IN_HZ* 定义的所需指定频率。

```
#define USE_CONN_IC              _FALSE
#define RUN_MODE               ANIMATION
#define IF_SEL                     SPI
```

宏 *USE_CONN_IC* 定义是否使用 UPAC 器件。当 LED 驱动器选择为 TLC696[2|3][3|6|9] 时，该宏会自动设置为 _TRUE。

宏 *RUN_MODE* 定义代码的运行模式。支持的运行模式包括动画模式和简单测试模式。

宏 *IF_SEL* 的选项为 *SPI*、*UART* 和 *UART_CAN*。对于 *UART* 和 *UART_CAN*，必须将 UPAC 器件选择宏 *USE_CONN_IC* 设置为 _TRUE。*UART* 与 *UART_CAN* 的区别在于：*UART* 在 MCU 和 UPAC 器件之间使用直接 UART 连接，而 *UART_CAN* 在 MCU 和 UPAC 器件之间使用 CAN 物理层。

以下代码块演示了会影响寄存器设置的宏。

```
#define TOTAL_SCAN_LINES       1
#define CASCADED_UNITS_BUS1    2
```

宏 *TOTAL_SCAN_LINES* 定义系统中使用的扫描线数。对于 TLC6962x/TLC6963x-Q1 系列，该值始终为 1。

宏 *CASCADED_UNITS_BUS1* 定义系统中级联器件的数量，并直接影响寄存器 DEVSET 中的字段 CHIP_NUM。对于 TLC69628Q1EVM 和 TLC69698Q1EVM，有两个器件级联；而对于 TLC69622Q1EVM 和 TLC69625Q1EVM，只有一个器件。

3.4 高级系统设置

为了说明高级系统设置，可参考图 3-2、图 3-3 和图 3-4 中所示的示例。

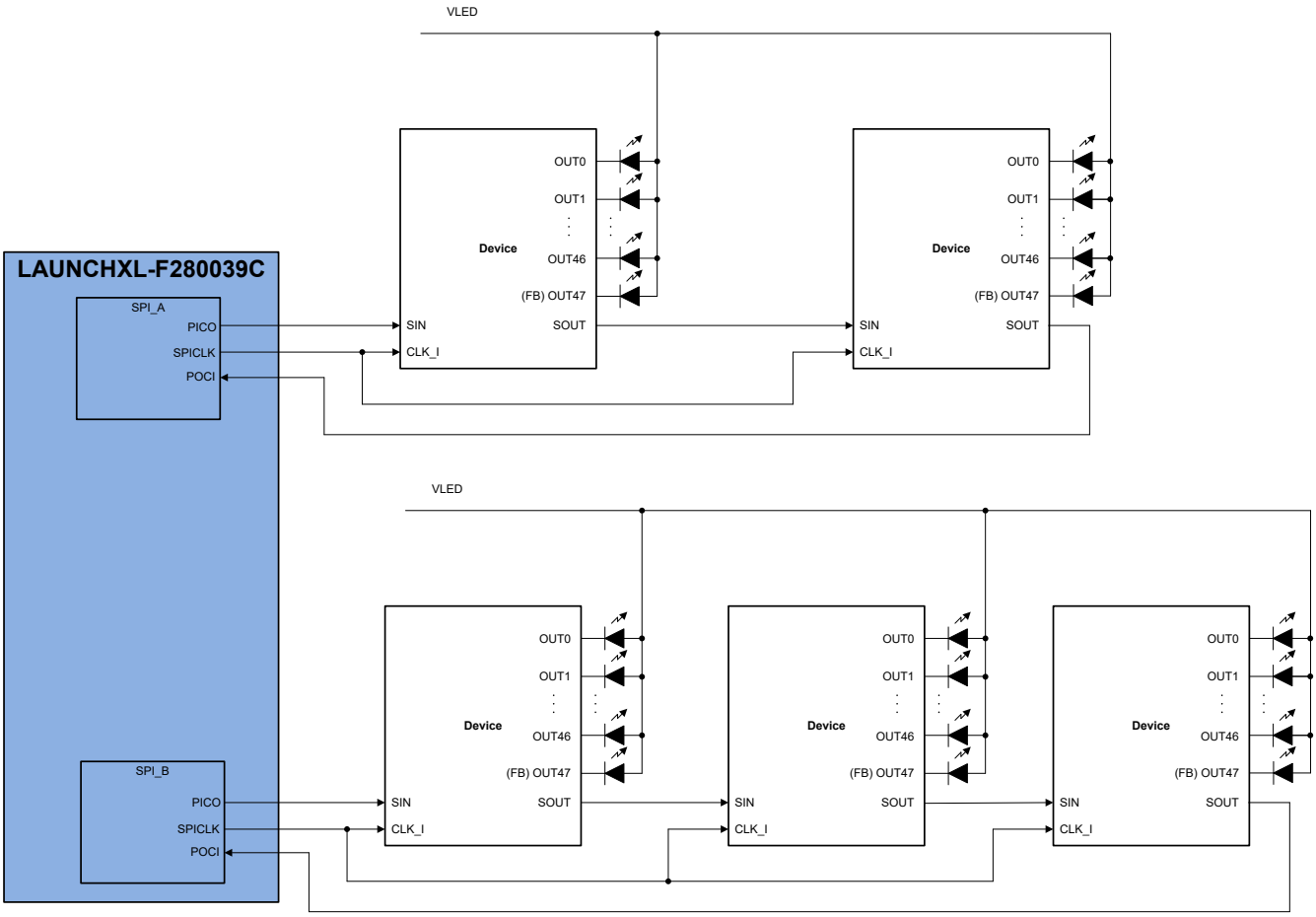

图 3-2. 具有 2 个菊花链且采用直接 SPI 连接的高级系统示例

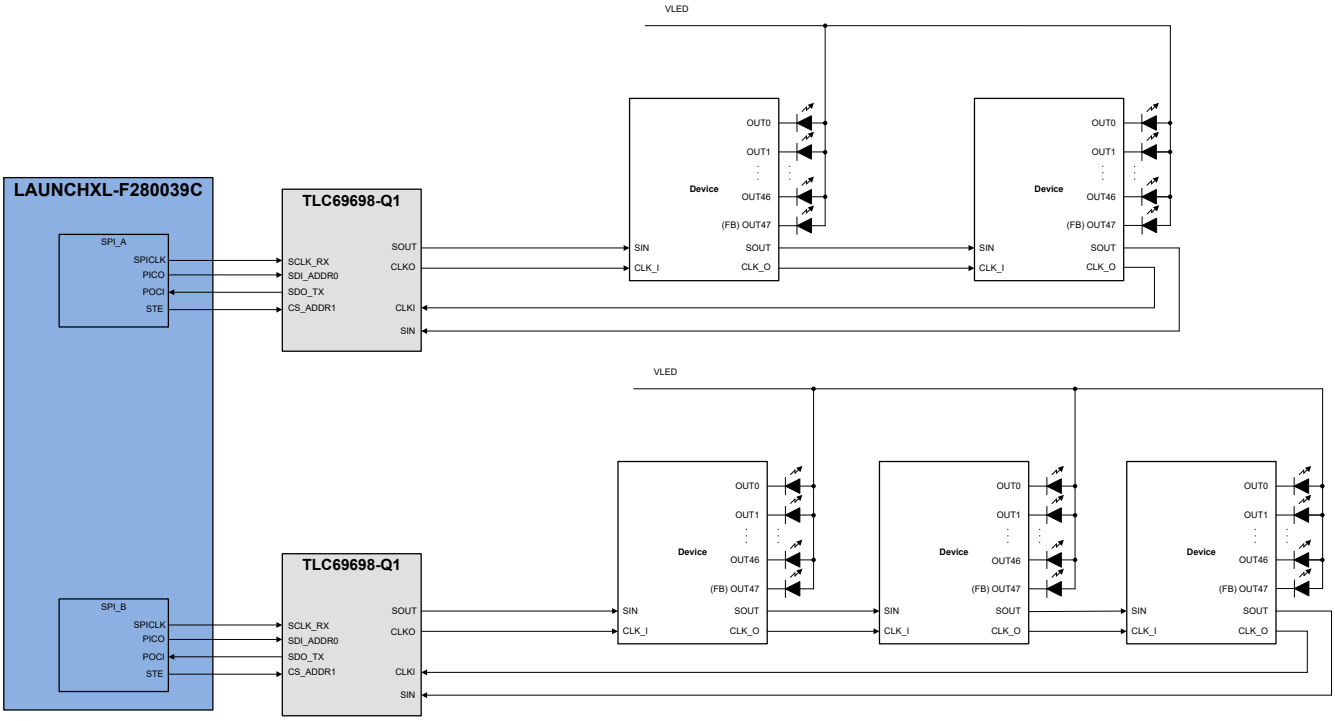

图 3-3. 具有 2 个菊花链且采用 SPI 连接至 UPAC 器件的高级系统示例

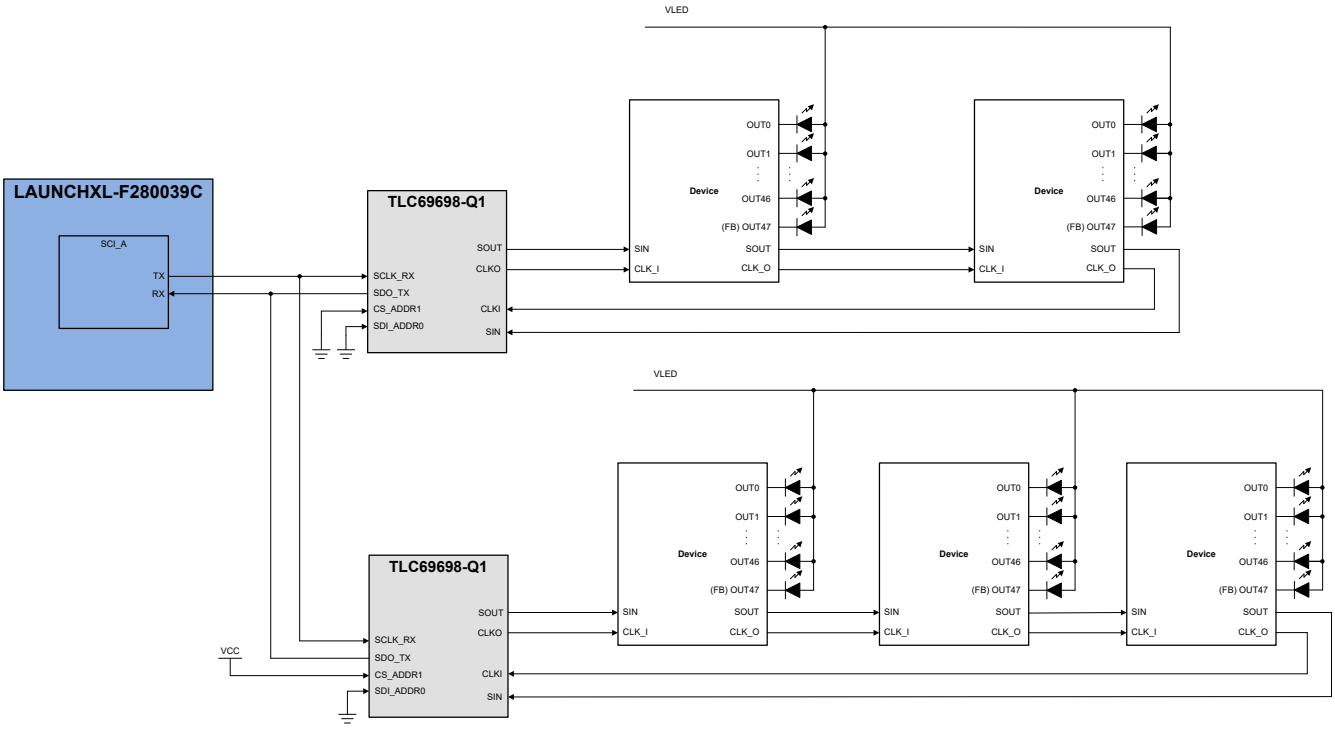

图 3-4. 具有 2 个菊花链且采用 UART 连接至 UPAC 器件的高级系统示例

使用 SPI 时，示例代码最多支持两个菊花链。对于 UART，限制为四个菊花链，该限制来自 TLC69698-Q1/TLC69697-Q1。实际使用的链数由文件 `system_info.h` 中的宏 *BUS_NUM* 定义。

```
// Total buses supported
#define BUS_NUM            2
```

每个链可具有不同数量的级联器件。因此，除了节 3.3 中描述的宏 *CASCADED_UNITS_BUS1* 之外，`system_info.h` 文件中还有宏 *CASCADED_UNITS_BUS2*、*CASCADED_UNITS_BUS3*（仅支持 UART）和 *CASCADED_UNITS_BUS4*（仅支持 UART）。在这些示例中，链 1 有两个级联器件，链 2 有三个级联器件。

```
#define CASCADED_UNITS_BUS1    2
#define CASCADED_UNITS_BUS2    3
```

请注意，对于图 3-4 中所示的示例，*BUS_NUM* 表示菊花链的数量。仍然仅支持一条 UART 总线。对于 UART，还必须设置 *dev_addr*。在该示例中，一个 TLC69698-Q1 的地址为 0，另一个 TLC69698-Q1 的地址为 2（SDI_ADDR0 拉低，CS_ADDR1 拉高）。地址的顺序必须与 CASCADED_UNITS_BUSx 的定义相匹配。

```
const uint16_t dev_addr[BUS_NUM] = {DEVICE_ADDR__0, DEVICE_ADDR__2};
```

3.5 诊断

示例代码提供了一个 API 来检测哪些 LED 驱动器器件具有故障，例如热关断 (TSD)、LED 开路 (LOD)、LED 短路 (LSD) 和相邻引脚短路 (APS)。对于 LOD、LSD、LED_CHK、DRV_CHK 和 APS，还会检测故障通道。此外，当宏 *USE_CONN_IC* 设置为 _TRUE 时，将读取 UPAC 器件检测到的故障。`TLC6962x_3x_APIs.h` 文件中定义了该 API 的原型。

```
void LED_Update_Chip_Status(void);
```

该 API 会更新 *chip_status* 变量（此变量在 `system_info.h` 文件中定义）。

```
struct busStatus {
    uint16_t chip_index;      // Show Chip Index
    uint16_t LOD;             // LED Open Detection
    uint16_t LSD;             // LED Short Detection
    uint16_t FB_OVF;          // Device FB overflow flag
    uint16_t COMM_ERR;        // Device communication error flag including CRC, COMM_LOSS, CMD_ERR,
TOUT_ERR, MEM_ERR
    uint16_t UVLO;            // Device UVLO flag
    uint16_t TSD;             // Thermal Shutdown
    uint16_t PIN_ERR;         // Device PIN error flag including APS, LED_CHK, DRV_CHK, ISET_O,
ISET_S
    uint16_t BIST;            // Device BIST error flag including WDOG_BIST, TOUT_BIST, OTP_CRC
    uint16_t DEV_MODE;        // Device mode
    uint16_t DEC;             // Device FB decrease flag
    uint16_t INC;             // Device FB increase flag
    uint16_t APS;             // Device Adjacent Pin Short flag
    uint16_t DRV_CHK;         // Device DRV_CHK flag
    uint16_t LED_CHK;         // Device LED_CHK flag
    uint16_t CRC;             // Device CRC flag
    uint16_t COMM_WDOG;       // Device COMM_WDOG flag
    uint16_t CMD_ERR;         // Device CMD_ERR flag
    uint16_t TOUT_ERR;        // Device time-out error fla
    uint16_t ISET_O;          // Device ISET open flag
    uint16_t ISET_S;          // Device ISET short
    uint16_t SRAM_CRC;        // Device MEM_ERR flag
    uint16_t OTP_CRC;         // Device OTP CRC error flag
    uint16_t WDOG_BIST;       // Device watchdog BIST error flag
    uint16_t TOUT_BIST;       // Device time-out BIST error flag
    volatile OUTx LSD_channels[MAX_SCAN_LINES];
    volatile OUTx LOD_channels[MAX_SCAN_LINES];
    volatile OUTx APS_channels[MAX_SCAN_LINES];
    volatile OUTx DRV_CHK_channels[MAX_SCAN_LINES];
    volatile OUTx LED_CHK_channels[MAX_SCAN_LINES];

};

struct chipStatus {
    struct busStatus busStatus[MAX_CASCADED_UNITS];
#ifdef _TLC69698
    uint16_t ERR;                     // Global error flag
    uint16_t POR;                     // Power-On-Reset flag
    uint16_t OTP_CRC;                 // OTP CRC error
    uint16_t DEV_STATE;               // Device state
    uint16_t CTRL_IF_CRC;             // SPI/UART CRC error
    uint16_t CTRL_IF;                 // SPI/UART error
    uint16_t CTRL_IF_TIMEOUT;         // SPI/UART reset timeout error
    uint16_t SPI_CS;                  // SPI Chip Select (CS) pin error
    uint16_t UART_NE;                 // UART Noise Error
    uint16_t UART_FE;                 // UART Frame Error
    uint16_t UART_RST;                // UART Reset
    uint16_t CSI_IDLE;                // CSI Peripheral in IDLE state
    uint16_t FIFO;                    // FIFO error
    uint16_t RXFFOVF;                 // Receive FIFO overflow
    uint16_t TXFFOVF;                 // Transmit FIFO overflow
    uint16_t DAISY_CHAIN;             // LED driver error
    uint16_t LOD;                     // Daisy Chain LOD flag
    uint16_t LSD;                     // Daisy Chain LSD flag
    uint16_t DRV_IF;                  // CSI interface error
    uint16_t CSI_COMM_WDOG;           // Device CSI communication watchdog flag
    uint16_t CSI_TOUT_ERR;            // Device CSI time-out error flag
    uint16_t CSI_CMD_ERR;             // Device CSI command error flag
    uint16_t CSI_CRC;                 // Device CSI CRC flag.
    uint16_t FLTSUM_ONGOING;          // Fault summary cycle in progress flag.
```

```
    uint16_t FLTSUM_DATAVALID;          // Fault summary data valid flag.
    uint16_t DEV_CNT_LOD_LSD;           // Device count that have LOD and/or LSD.
    uint16_t ZONE_CNT_LOD_LSD;          // Total zone count that have LOD and/or LSD.
    uint16_t SZ_CNT_LOD_LSD;            // Total safety zones channel count that have LOD and/or LSD.
#endif
};

// For diagnostics
extern struct chipStatus chip_status[BUS_NUM];
```

在代码调试期间，可以按照观察变量、表达式和寄存器中的步骤在表达式视图中观察变量 *chip_status*。图 3-5 中描述了一个没有任何错误的示例。变量 *chip_status* 的第一个索引是菊花链索引。在 EVM 上只有一个链。对于 LED 驱动器的标志，使用 busStatus 变量，该变量具有链中每个 LED 驱动器的索引。在 TLC69628Q1EVM 和 TLC69698Q1EVM 上有 2 个级联的器件，因此 busStatus 索引的编号是从 0 到 1。对于 TLC69622Q1EVM 和 TLC69625Q1EVM，只有 1 个器件级联，因此 busStatus 只有索引 0。

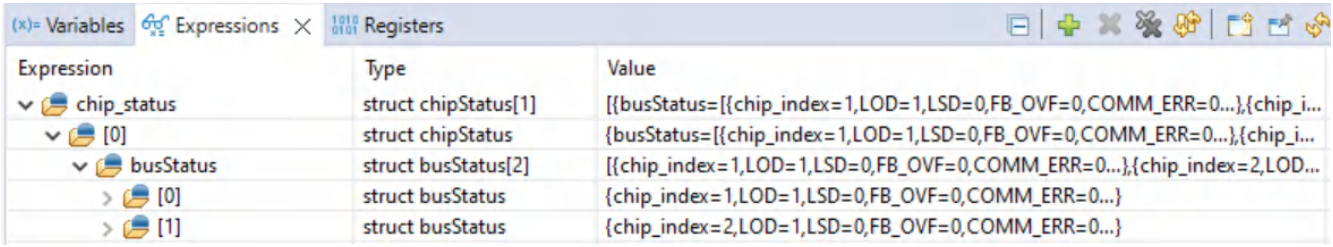

图 3-5. 观察表达式 chip_status（无错误）的示例

(x)= Variables | Expressions | Registers

Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0....
[0]	struct chipStatus	{busStatus=[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0.....
busStatus	struct busStatus[2]	[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_ind...
[0]	struct busStatus	{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...}
chip_index	unsigned int	1
LOD	unsigned int	0
LSD	unsigned int	0
FB_OVF	unsigned int	0
COMM_ERR	unsigned int	0
UVLO	unsigned int	0
TSD	unsigned int	0
PIN_ERR	unsigned int	0
BIST	unsigned int	0
DEV_MODE	unsigned int	0x0200 (Hex)
DEC	unsigned int	0
INC	unsigned int	0
APS	unsigned int	0
DRV_CHK	unsigned int	0
LED_CHK	unsigned int	0
CRC	unsigned int	0
COMM_WDOG	unsigned int	0
CMD_ERR	unsigned int	0
TOUT_ERR	unsigned int	0
ISET_O	unsigned int	0
ISET_S	unsigned int	0
SRAM_CRC	unsigned int	0
OTP_CRC	unsigned int	0
WDOG_BIST	unsigned int	0
TOUT_BIST	unsigned int	0
LSD_channels	union <unnamed>[1]	[{OUT=0x0000000000000000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0...
LOD_channels	union <unnamed>[1]	[{OUT=0x0000000000000000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0...
APS_channels	union <unnamed>[1]	[{OUT=0x0000000000000000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0...
DRV_CHK_channels	union <unnamed>[1]	[{OUT=0x0000000000000000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0...
LED_CHK_channels	union <unnamed>[1]	[{OUT=0x0000000000000000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0...
[1]	struct busStatus	{chip_index=2,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...}
chip_index	unsigned int	2
LOD	unsigned int	0
LSD	unsigned int	0
FB_OVF	unsigned int	0

图 3-6. 观察表达式 chip_status（无错误）的扩展示例

图 3-6 显示了 *chip_status* 变量的扩展视图，对于菊花链中的两个 LED 驱动器，每一个都显示了芯片索引和故障状态位。此外，对于 LSD、LOD、LED_CHK、DRV_CHK 和 APS 故障，可以找到具有故障的实际输出通道。LOD 故障的示例如图 3-7 所示。在此示例中，芯片索引 1（即 busStatus 的索引 0）具有 LOD 故障。数组 LOD_channels 扩展后，会显示故障发生在引脚 OUT12 上。LOD_channels 是一个仅有 1 个索引的数组，因为对于 TLC6962x/TLC6963x-Q1 器件，只有 1 条扫描线。

(x)= Variables | Expressions | Registers

Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0....
[0]	struct chipStatus	{busStatus=[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0.....
busStatus	struct busStatus[2]	[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_ind...
[0]	struct busStatus	{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...}
chip_index	unsigned int	1
LOD	unsigned int	1
LSD	unsigned int	0
FB_OVF	unsigned int	0
COMM_ERR	unsigned int	0
UVLO	unsigned int	0
TSD	unsigned int	0
PIN_ERR	unsigned int	0
BIST	unsigned int	0
DEV_MODE	unsigned int	0x0200 (Hex)
DEC	unsigned int	0
INC	unsigned int	0
APS	unsigned int	0
DRV_CHK	unsigned int	0
LED_CHK	unsigned int	0
CRC	unsigned int	0
COMM_WDOG	unsigned int	0
CMD_ERR	unsigned int	0
TOUT_ERR	unsigned int	0
ISET_O	unsigned int	0
ISET_S	unsigned int	0
SRAM_CRC	unsigned int	0
OTP_CRC	unsigned int	0
WDOG_BIST	unsigned int	0
TOUT_BIST	unsigned int	0
LSD_channels	union <unnamed>[1]	[{OUT=0x0000000000000000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0...
LOD_channels	union <unnamed>[1]	[{OUT=0x0000000000001000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0...
[0]	union <unnamed>	{OUT=0x0000000000001000,PT0={OUT0=0x0,OUT1=0x0,OUT2=0x0,...
OUT	unsigned long long	0x0000000000001000 (Hex)
PT0	struct <unnamed>	{OUT0=0x0,OUT1=0x0,OUT2=0x0,OUT3=0x0,OUT4=0x0...} (Hex)
OUT0	unsigned int : 1	0x0 (Hex)
OUT1	unsigned int : 1	0x0 (Hex)
OUT2	unsigned int : 1	0x0 (Hex)
OUT3	unsigned int : 1	0x0 (Hex)
OUT4	unsigned int : 1	0x0 (Hex)
OUT5	unsigned int : 1	0x0 (Hex)
OUT6	unsigned int : 1	0x0 (Hex)
OUT7	unsigned int : 1	0x0 (Hex)
OUT8	unsigned int : 1	0x0 (Hex)
OUT9	unsigned int : 1	0x0 (Hex)
OUT10	unsigned int : 1	0x0 (Hex)
OUT11	unsigned int : 1	0x0 (Hex)
OUT12	unsigned int : 1	0x1 (Hex)
OUT13	unsigned int : 1	0x0 (Hex)

图 3-7. 具有 LED 开路 (LOD) 故障的 chip_status 示例

当 LED 驱动器与 UPAC 器件配对使用时，*chip_status* 变量还会显示 UPAC 器件的错误标志。图 3-8 中描述了一个没有任何故障的示例，其中为每个菊花链（*chip_status* 的第一个索引）列出了错误标志。请注意，CSI_IDLE 为 1，因为在诊断期间，UPAC 器件不转发任何消息。FLTSUM_DATAVALID 为 1，因为示例代码中的诊断发生在 TLC69698-Q1 的故障汇总第一个周期之后。

(x)= Variables | Expressions | Registers

Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0....
[0]	struct chipStatus	{busStatus=[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0.....
busStatus	struct busStatus[2]	[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_ind...
[0]	struct busStatus	{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...}
[1]	struct busStatus	{chip_index=2,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...}
ERR	unsigned int	0
POR	unsigned int	0
OTP_CRC	unsigned int	0
DEV_STATE	unsigned int	0x0020 (Hex)
CTRL_IF_CRC	unsigned int	0
CTRL_IF	unsigned int	0
CTRL_IF_TIMEOUT	unsigned int	0
SPI_CS	unsigned int	0
UART_NE	unsigned int	0
UART_FE	unsigned int	0
UART_RST	unsigned int	0
CSI_IDLE	unsigned int	1
FIFO	unsigned int	0
RXFFOVF	unsigned int	0
TXFFOVF	unsigned int	0
DAISY_CHAIN	unsigned int	0
LOD	unsigned int	0
LSD	unsigned int	0
DRV_IF	unsigned int	0
CSI_COMM_WDOG	unsigned int	0
CSI_TOUT_ERR	unsigned int	0
CSI_CMD_ERR	unsigned int	0
CSI_CRC	unsigned int	0
FLTSUM_ONGOING	unsigned int	0
FLTSUM_DATAVALID	unsigned int	1
DEV_CNT_LOD_LSD	unsigned int	0
ZONE_CNT_LOD_LSD	unsigned int	0
SZ_CNT_LOD_LSD	unsigned int	0

图 3-8. 与 TLC69698-Q1 配对使用时未出现错误的 chip_status 示例

使用 UPAC 器件时发生 LOD 故障的示例如图 3-9 所示。在此示例中，TLC69698-Q1 中的 DAISY_CHAIN 和 LOD 标志均被置位。此外，TLC69698-Q1 的故障汇总功能统计出有 1 个器件存在故障 (DEV_CNT_LOD_LSD)，并且有 1 个 LED 区域存在故障 (ZONE_CNT_LOD_LSD)。发生故障的 LED 区域未被识别为 LED 驱动器中的所谓安全区域。因此，SZ_CNT_LOD_LSD 仍为 0。

(x)= Variables | Expressions | Registers

Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0....
[0]	struct chipStatus	{busStatus=[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0.....
busStatus	struct busStatus[2]	[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_ind...
[0]	struct busStatus	{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...}
[1]	struct busStatus	{chip_index=2,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...}
ERR	unsigned int	1
POR	unsigned int	0
OTP_CRC	unsigned int	0
DEV_STATE	unsigned int	0x0020 (Hex)
CTRL_IF_CRC	unsigned int	0
CTRL_IF	unsigned int	0
CTRL_IF_TIMEOUT	unsigned int	0
SPI_CS	unsigned int	0
UART_NE	unsigned int	0
UART_FE	unsigned int	0
UART_RST	unsigned int	0
CSI_IDLE	unsigned int	1
FIFO	unsigned int	0
RXFFOVF	unsigned int	0
TXFFOVF	unsigned int	0
DAISY_CHAIN	unsigned int	1
LOD	unsigned int	1
LSD	unsigned int	0
DRV_IF	unsigned int	0
CSI_COMM_WDOG	unsigned int	0
CSI_TOUT_ERR	unsigned int	0
CSI_CMD_ERR	unsigned int	0
CSI_CRC	unsigned int	0
FLTSUM_ONGOING	unsigned int	0
FLTSUM_DATAVALID	unsigned int	1
DEV_CNT_LOD_LSD	unsigned int	1
ZONE_CNT_LOD_LSD	unsigned int	1
SZ_CNT_LOD_LSD	unsigned int	0

图 3-9. 与 TLC69698-Q1 配对使用时具有 LOD 故障的 Expression chip_status 示例

在图 3-10 所示的示例中，故障汇总统计出 2 个器件中有 3 个 LED 故障，其中所有 3 个 LED 区域都被指定为所谓的安全区域。当安全区域发生故障时，TLC69698-Q1 还会记录故障位置。这些位置可以在变量 *fault_summary* 中找到，如图 3-11 所示。最后一个有效条目之后的条目填充为 0xFFFF，以表示数组结束。这意味着，在此示例中，数组的索引 3 填充为 0xFFFF，因为索引 0 到 2 包含了发生故障的 LED 安全区域的有效位置。

(x)= FLTSUM_ONGOING	unsigned int	0
(x)= FLTSUM_DATAVALID	unsigned int	1
(x)= DEV_CNT_LOD_LSD	unsigned int	2
(x)= ZONE_CNT_LOD_LSD	unsigned int	3
(x)= SZ_CNT_LOD_LSD	unsigned int	3

图 3-10. 与 TLC69698-Q1 配对使用时具有 3 个故障的表达式 chip_status 示例

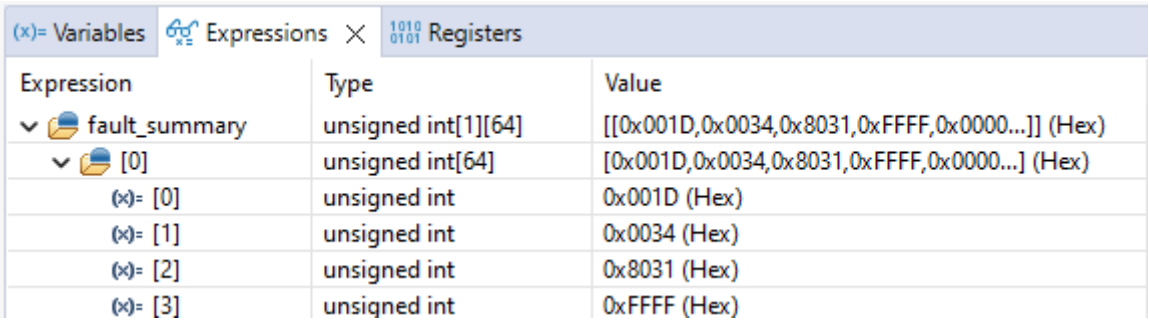

图 3-11. 与 TLC69698-Q1 配对使用时具有 3 个故障的表达式 fault_summary 示例

3.6 演示

本节介绍了运行演示的示例。图 3-12 显示了在动画模式下运行的 TLC69628Q1EVM。图 3-13 展示了 TLC69698Q1EVM，图 3-14 展示了 TLC69622Q1EVM，图 3-15 展示了 TLC69625Q1EVM。

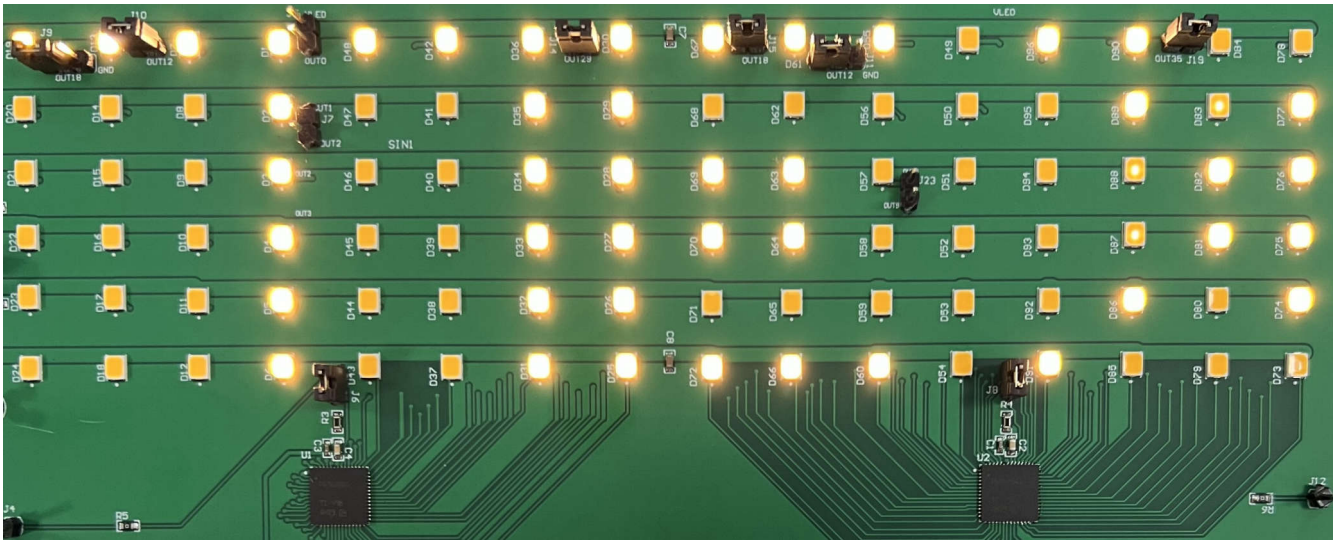

图 3-12. TLC69628Q1EVM 的示例演示

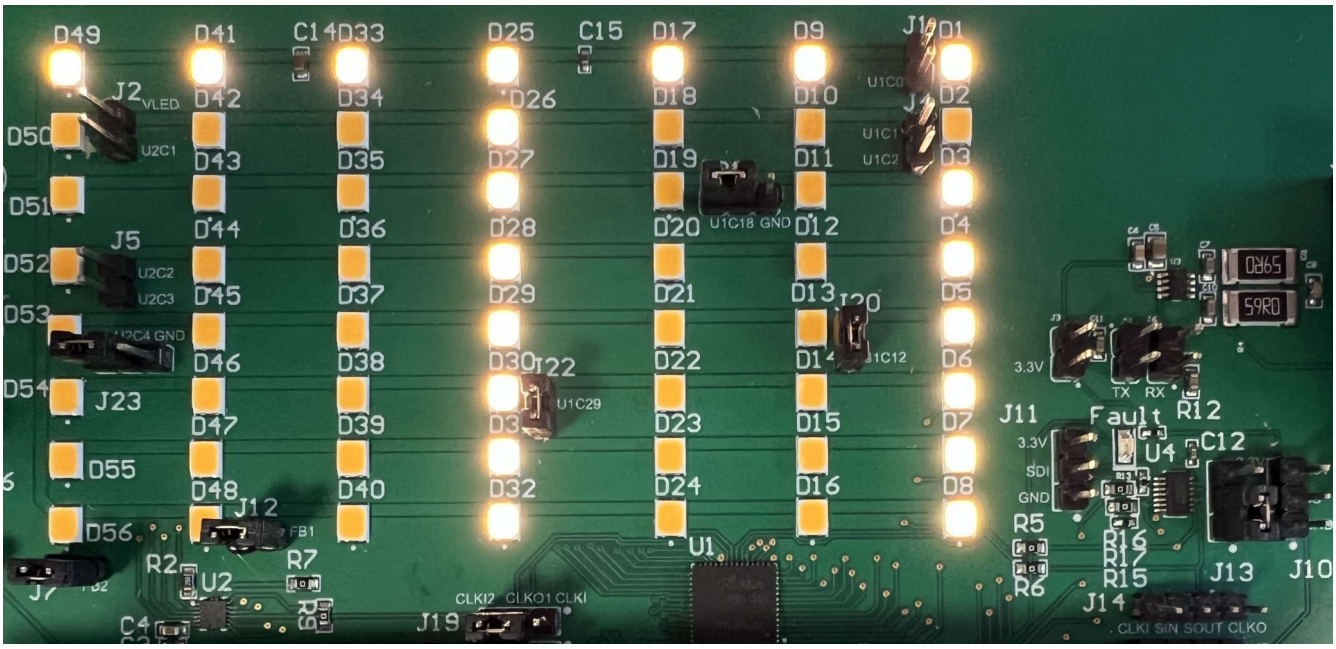

图 3-13. TLC69698Q1EVM 的示例演示

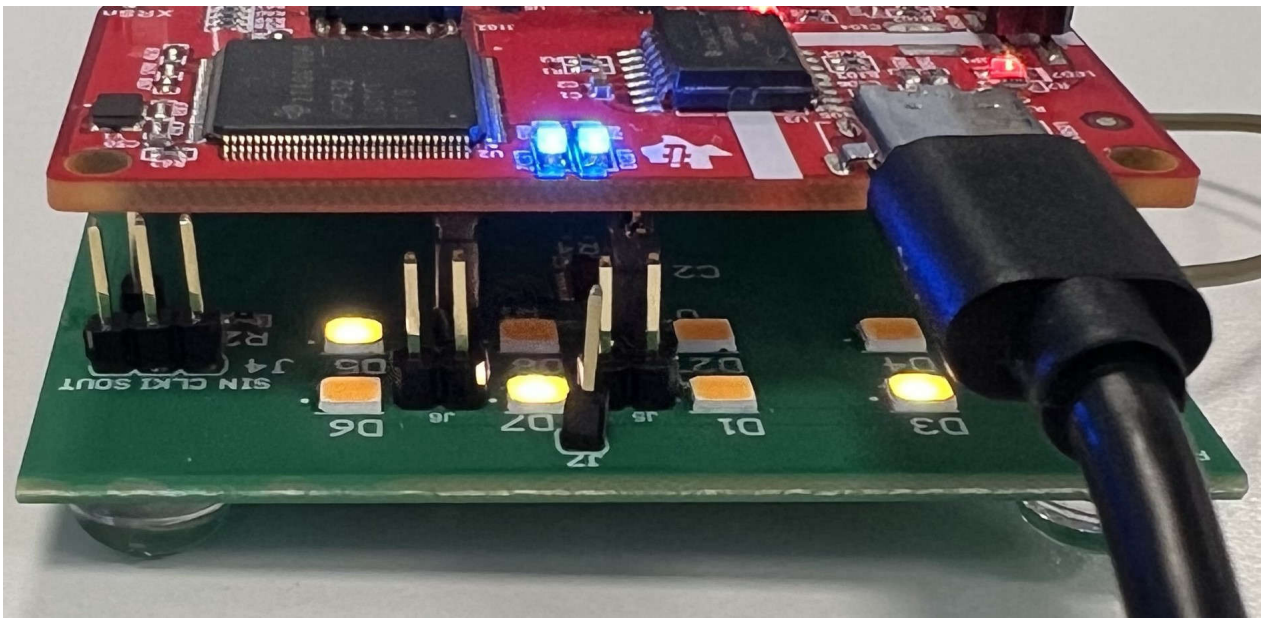

图 3-14. TLC69622Q1EVM 的示例演示

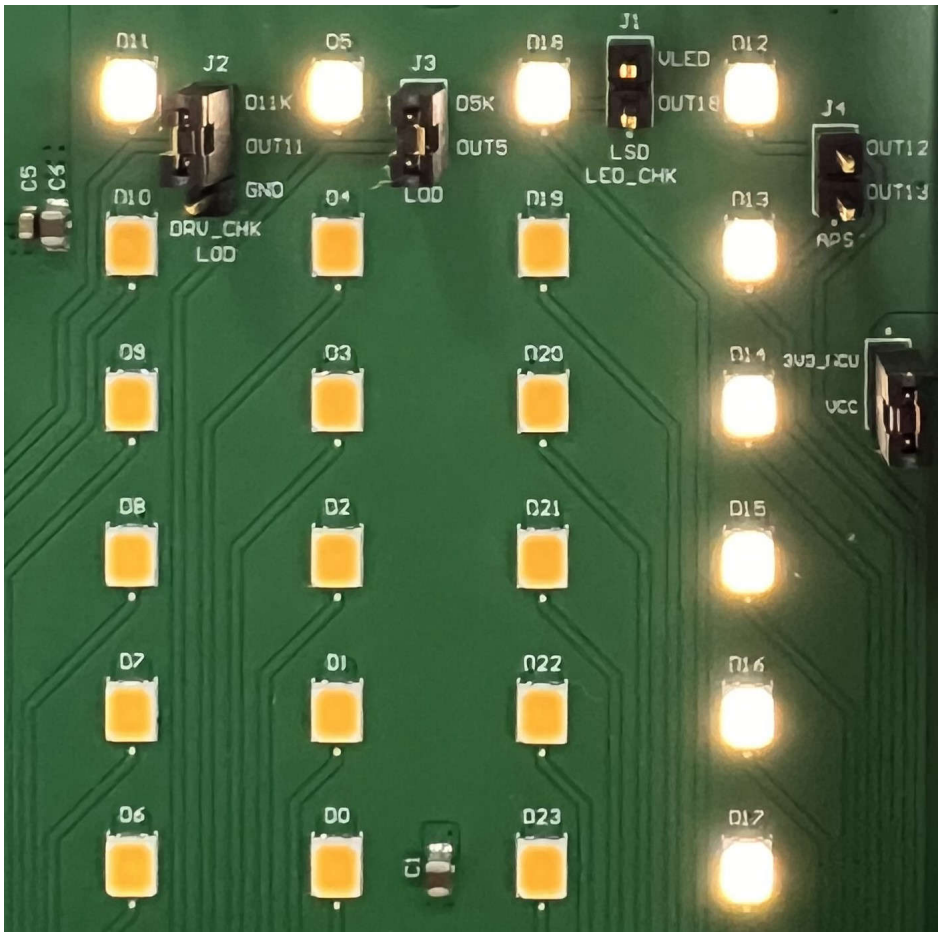

图 3-15. TLC69625Q1EVM 的示例演示

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

Changes from October 31, 2025 to March 31, 2026 (from Revision A (October 2025) to Revision B (March 2026)) **Page**

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