User's Guide

LP589x(-Q1)/TLC698x 示例代码

摘要

本文档介绍了准备和使用与 LAUCHXL-F280039C 配对使用时的 LP589x(-Q1)/TLC698x 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。LP5890/1/2(-Q1) 器件可与 LP5899(-Q1) 配对使用，以使用 MCU 的标准 SPI 模块与 LED 驱动器通信。此外，TLC6983/4 器件可与 TLC6989 配对使用，以使用 MCU 的标准 SPI 模块与 LED 驱动器通信。

内容

1 引言 2
2 软件设置 3
3 示例代码结构 5
3.1 设计参数 5
3.2 流程图 5
3.3 系统设置 7
3.4 诊断 10
3.5 演示 14
4 修订历史记录 15

插图清单

图 2-1. Code Composer Studio 安装过程 3
图 2-2. C2000Ware 5.1.0.00 安装验证 4
图 3-1. 示例代码流程图 6
图 3-2. 不使用增强连接 IC、使用 2 个 CCSI 链的示例系统 8
图 3-3. 使用增强连接 IC 和 2 个 CCSI 链的示例系统 9
图 3-4. 观察表达式 chip_status (无错误) 的示例 11
图 3-5. 显示具有 LOD 的 chip_status 表达式的扩展扫描线的示例 11
图 3-6. 显示具有 LOD 的 chip_status 表达式的扩展通道的示例 12
图 3-7. 与 LP5899 配对使用时未出现错误的 chip_status 示例 13
图 3-8. 与 LP5899 配对使用时具有 CCSI 故障的 Expression chip_status 示例 14
图 3-9. LP5891Q1EVM 演示示例 14
图 3-10. TLC6984EVM 演示示例 15

表格清单

表 3-1. 设计参数 EVM 5
表 3-2. 每个文件的宏和变量名称的摘要 7

商标

1 引言

示例代码展示了点亮 LP5891Q1EVM、LP5891EVM、LP5890EVM、TLC6984EVM 和 TLC6983EVM 上的 LED 的功能。每个 EVM 都有各自的示例代码。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，两个帧用于向左、向右、向上和向下滚动，并根据预定义的顺序淡入淡出。第一个帧是德州仪器 (TI) 32x32 RGB 像素的标识，第二个帧是 48x32 RGB 像素的彩虹图案。这意味着 EVM 的 LED 显示屏上并不总是显示整个帧。有关这方面的示例，请参阅节 3.5。本文档不会解释如何生成示例代码中的帧。

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连续时钟串行接口 (CCSI) 命令。示例代码附带打开所有 RGB LED 的功能，显示屏会显示为白色。此外，对每个帧都会执行诊断。有关诊断的更多信息，请参阅节 3.4。

LP5891Q1EVM、LP5891EVM 和 LP5890EVM 可与 LP5899DYYEVM 配对使用。节 3.3 介绍了如何启用配对。启用后，使用 LAUCHXL-F280039C 的 SPI 与 LP5899DYYEVM 进行通信，而不是使用 LAUCHXL-F280039C 的 CLB（可配置逻辑块）与 LP589X(Q1) EVM 进行通信。此外，还支持额外诊断功能。

示例代码中预定义的 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。

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™。
   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。

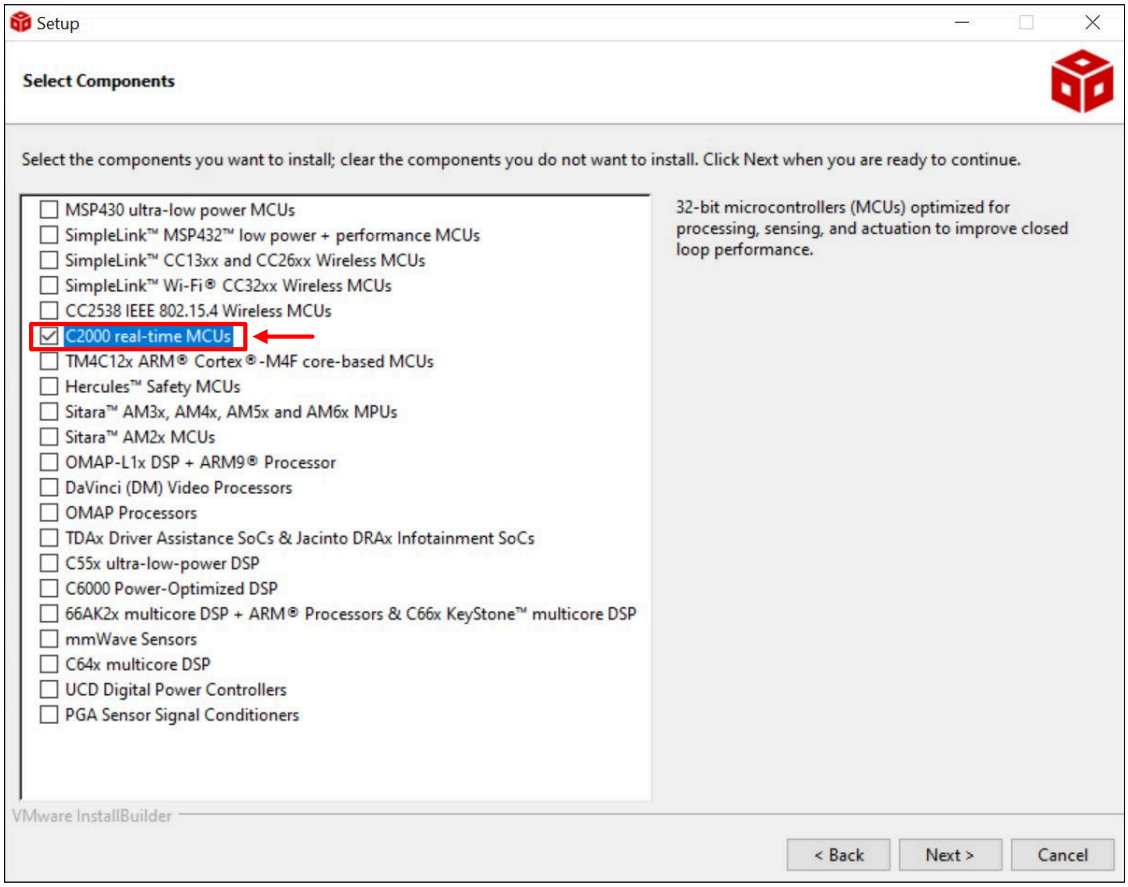

图 2-1. Code Composer Studio 安装过程

2. 下载并安装 C2000Ware（版本 5.01.00.00）。
   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 5.1.0.00”和“SysConfig”（安装 C2000Ware 时应自动安装 SysConfig），如图 2-2 所示。如果没有出现“C2000Ware 5.1.0.00”或“SysConfig”，您可能需要点击“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。

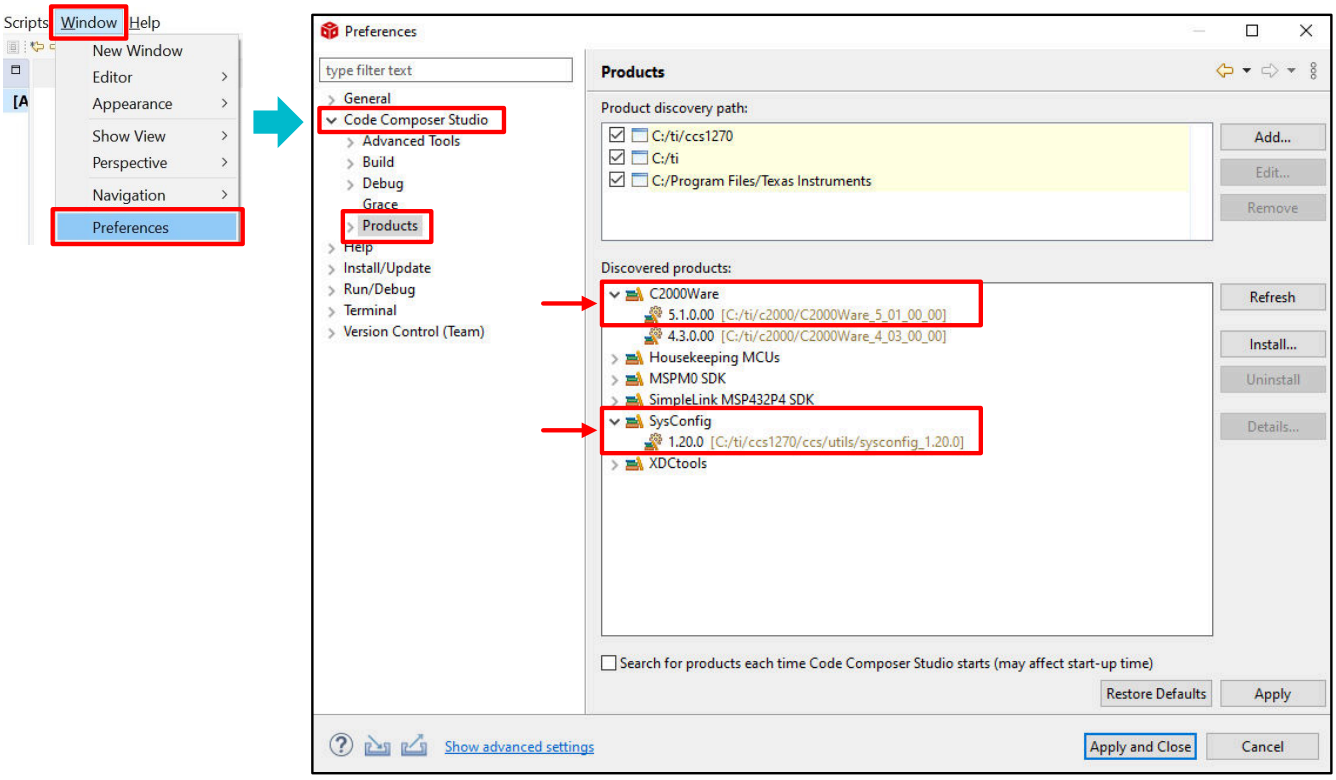

图 2-2. C2000Ware 5.1.0.00 安装验证

3. 下载并导入示例代码。
   a. 每个 EVM 的链接不同。但是，除了为匹配 EVM 设置的 `led_driver.h` 和 `system_info.h` 文件之外，每个链接中的示例代码都是相同的。
      i. LP5891Q1EVM：LP5891Q1EVM-F280039C-SW
      ii. LP5891EVM：LP5891Q1EVM-F280039C-SW
      iii. LP5890EVM：LP5890EVM-F280039C-SW
      iv. TLC6984EVM：TLC6984EVM-F280039C-SW
      v. TLC6983EVM：TLC6983EVM-F280039C-SW
      vi. LP5899DYYEVM 与 LP5891(Q1)EVM 配对使用：LP5899DYYEVM-LP5891-F280039C-SW
      vii. LP5899DYYEVM 与 LP5890EVM 配对使用：LP5899DYYEVM-LP5890-F280039C-SW
   b. 根据链接中提供的过程导入 Code Composer Studio (CCS) 工程：导入 CCS 工程。
4. 根据链接中提供的过程加载程序：构建和运行工程。
5. （可选）下载寄存器映射生成工具。如果您想进一步配置寄存器，这是一个方便的工具。
   a. LP5891-Q1：LP5891 寄存器映射生成工具
   b. LP5891：LP5891 寄存器映射生成工具
   c. LP5890：LP5890 寄存器映射生成工具
   d. TLC6984：TLC6984 寄存器映射生成工具
   e. TLC6983：TLC6983 寄存器映射生成工具

3 示例代码结构

3.1 设计参数

表 3-1 中列出了用于不同 EVM 的 LED 矩阵显示设计参数。

表 3-1. 设计参数 EVM

设计参数	LP589x	TLC698x
显示模块大小	16 × 16 RGB LED	32 × 32 RGB LED
帧速率	60Hz	25Hz
刷新率	7680Hz	2400Hz
PWM 分辨率	16 位	16 位
级联器件	1	2
SCLK 频率	10MHz	10MHz
GCLK 频率	80MHz	120MHz

3.2 流程图

图 3-1 展示了示例代码中的大概流程。

在设置 MCU 期间，文件 `system_info.h` 中定义的几个宏确定是使用 SPI 与增强型连接器件通信，还是使用可配置逻辑块 (CLB) 通过 CCSI 与 LED 驱动器通信。当使用 SPI 进行通信时，增强型连接器件会针对 CCSI 数据速率和 FIFO 级别进行初始化。

`FC_settings.h` 文件可由节 2 中提到的寄存器映射生成工具自动生成。并非所有寄存器设置都来自该文件。扫描线数（寄存器 FC0 中的字段 SCAN_NUM）和级联器件的数量（寄存器 FC0 中的字段 CHIP_NUM）来自文件 `system_info.h`。节 3.3 更详细地介绍了此文件。

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 2 个帧。本文档未说明如何生成这些帧。

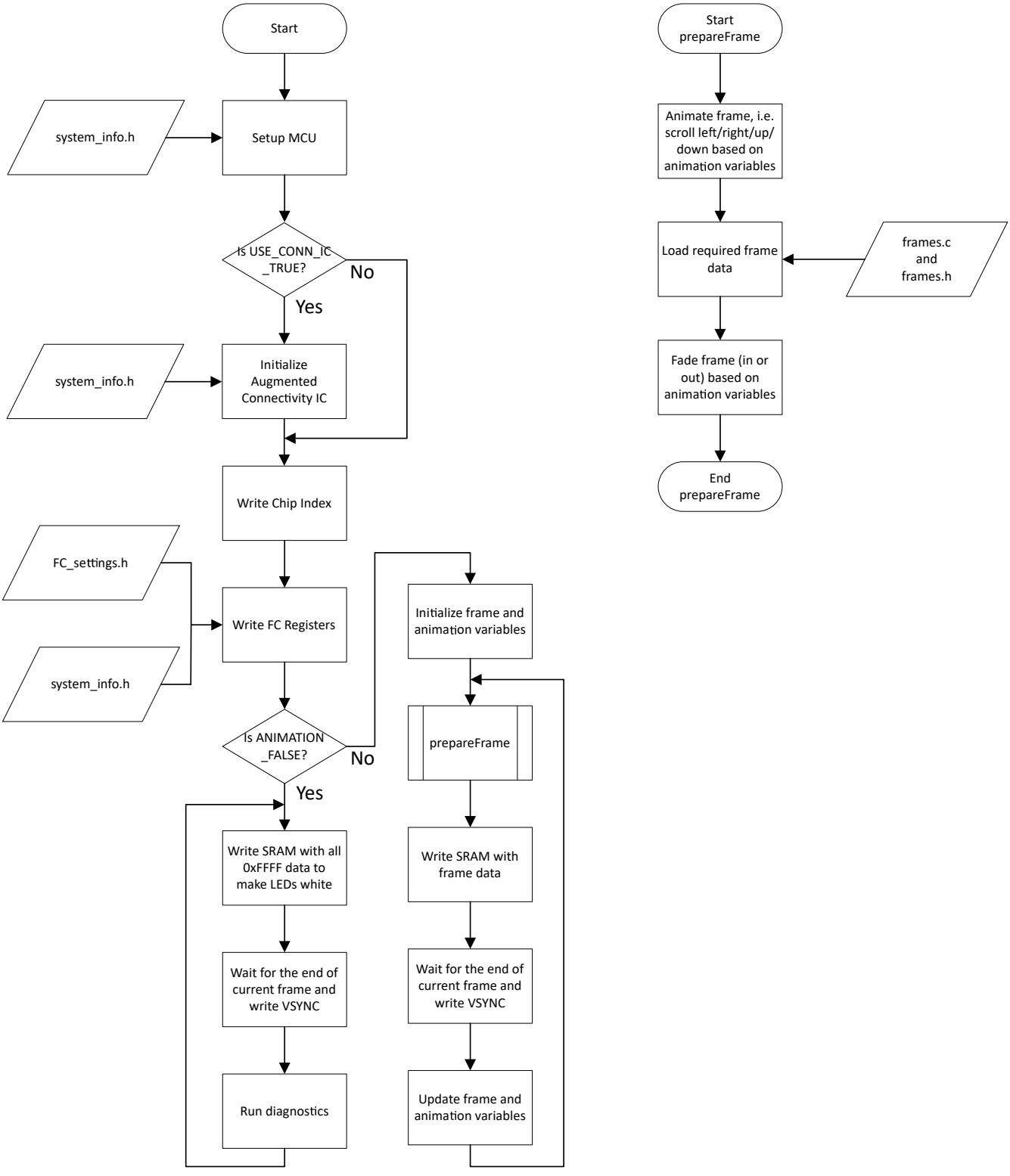

图 3-1. 示例代码流程图

3.3 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，选择了使用过的 LED 驱动器 IC。

```
#include "LP5891.h"
```

该代码支持：

- LP5892
- LP5891
- LP5890
- TLC6984
- TLC6983

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及其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	*SCLK_FREQ_IN_HZ*	SCLK 频率
	USE_CONN_IC	在使用增强型连接器件 LP5899(-Q1)/TLC6989 时选择
	SPI_FREQ_IN_HZ	使用增强型连接器件时，这是 MCU 与连接器件之间的 SPI 频率
	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*
	TOTAL_SCAN_LINES	扫描线数
	CCSI_BUS_NUM	CCSI 总线数
	CASCADED_UNITS_CCSI1	CCSI 总线 1 中的器件数量
	CASCADED_UNITS_CCSI2	CCSI 总线 2 中的器件数量
	MONOCHROMATIC	在 RGB 和单色显示屏之间进行选择
system_info.c	*FRAME_RATE*	VSYNC 命令的间隔

帧速率在文件 system_info.c 中指定。帧速率以 Hz (每秒帧数) 为单位。

```
const uint16_t FRAME_RATE = 60;      // 16.67ms = 60 Hz frames-per-second
```

受支持的最小帧速率为 1Hz。

文件 system_info.h 包含多个系统定义。

```
// When TLC6989 or LP5899 is used this should be set to _TRUE
#define USE_CONN_IC           _TRUE
```

宏 *USE_CONN_IC* 定义 LED 驱动器是否与增强型连接 IC 配对使用。这会影响 MCU 的硬件设置，该设置会在标准 SPI (使用连接 IC 时) 或 CLB (使用 CCSI 直接与 LED 驱动器通信时) 之间自动切换。

```
// Desired SCLK frequency (in case of TLC698x this SCLK frequency is half of this)
// Note: Exact frequency may not be possible
#define SCLK_FREQ_IN_HZ        10000000
```

宏 *SCLK_FREQ_IN_HZ* 定义连续时钟串行接口 (CCSI) 以什么数据速率运行，即引脚 SCLK 的时钟频率。使用增强连接 IC 时，所选数据速率将是等于所需数据速率的选项或第一个高于所需数据速率的可用选项。不使用增强连接 IC 时，所需的 SCLK 频率是相对于 MCU 系统频率的整数分频值。在这两种情况下，实际 CCSI 频率可能不同于 *SCLK_FREQ_IN_HZ* 定义的所需指定频率。

```
// Desired SPI frequency - for TLC6989 or LP5899
// Supported range is 500kHz to 7.5MHz --> For higher frequencies need to enable SPI high speed mode
// Note: Exact frequency may not be possible
#define SPI_FREQ_IN_HZ          7500000
```

宏 *SPI_FREQ_IN_HZ* 仅在使用增强连接 IC 并定义所需的 SPI 频率时使用。该频率是 MCU 系统频率的整数分频值。因此，实际 SPI 频率可能不同于 *SPI_FREQ_IN_HZ* 定义的所需指定频率。

```
#define RUN_MODE                ANIMATION
#define MONOCHROMATIC           _FALSE
```

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

EVM 都使用 RGB LED。因此，宏 *MONOCHROMATIC* 定义为 _FALSE。示例代码确实支持使用单色 LED (例如仅使用红色 LED) 的系统。在此类情况下，宏 *MONOCHROMATIC* 应定义为 _TRUE。这会自动更改帧数据结构、动画算法和 API。

以下代码块演示了会影响寄存器设置的宏。

```
#define TOTAL_SCAN_LINES        16
#define CASCADED_UNITS_CCSI1    1
```

宏 *TOTAL_SCAN_LINES* 定义了系统中使用的扫描线数，并直接影响寄存器 FC0 中的字段 SCAN_NUM。对于 LP5891Q1EVM、LP5891EVM 和 LP5890EVM，有 16 条扫描线。对于 TLC6983EVM 和 TLC6984EVM，有 32 条扫描线。

宏 *CASCADED_UNITS_CCSI1* 定义系统中级联器件的数量，并直接影响寄存器 FC0 中的字段 CHIP_NUM。对于 LP5891Q1EVM、LP5891EVM 和 LP5890EVM，仅有 1 个器件级联。当用户使用可用连接器级联更多这类 EVM 时，必须更新此宏。

对于 TLC6983EVM 和 TLC6984EVM，一个 EVM 上有 2 个级联器件。

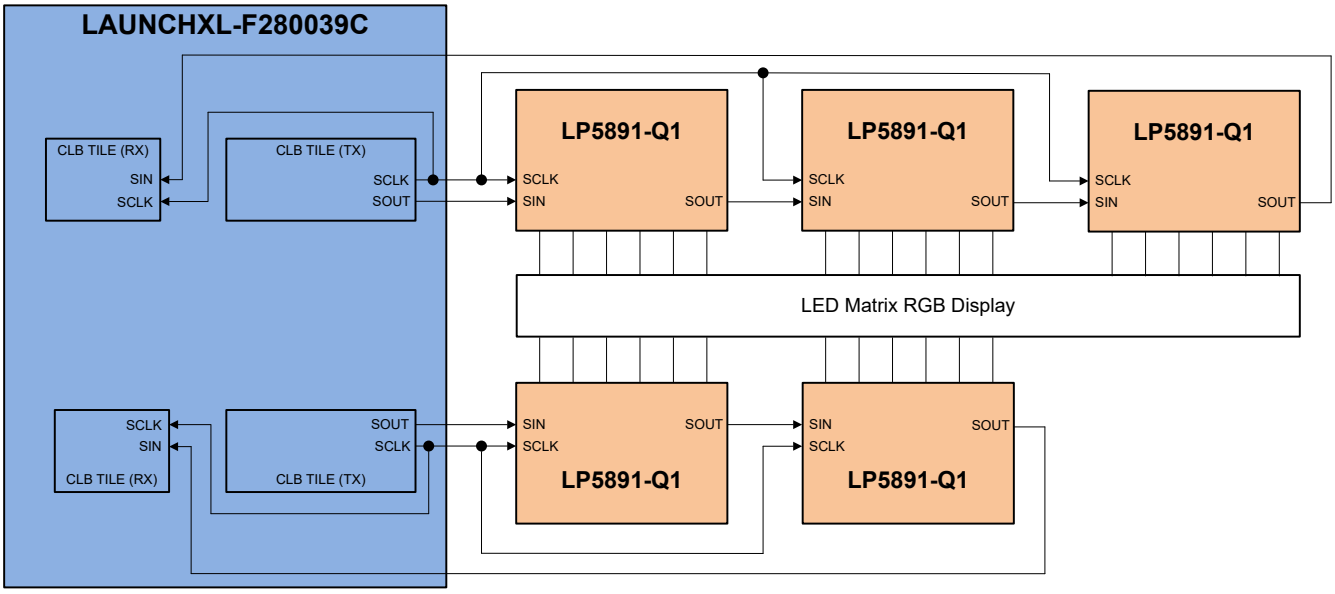

图 3-2. 不使用增强连接 IC、使用 2 个 CCSI 链的示例系统

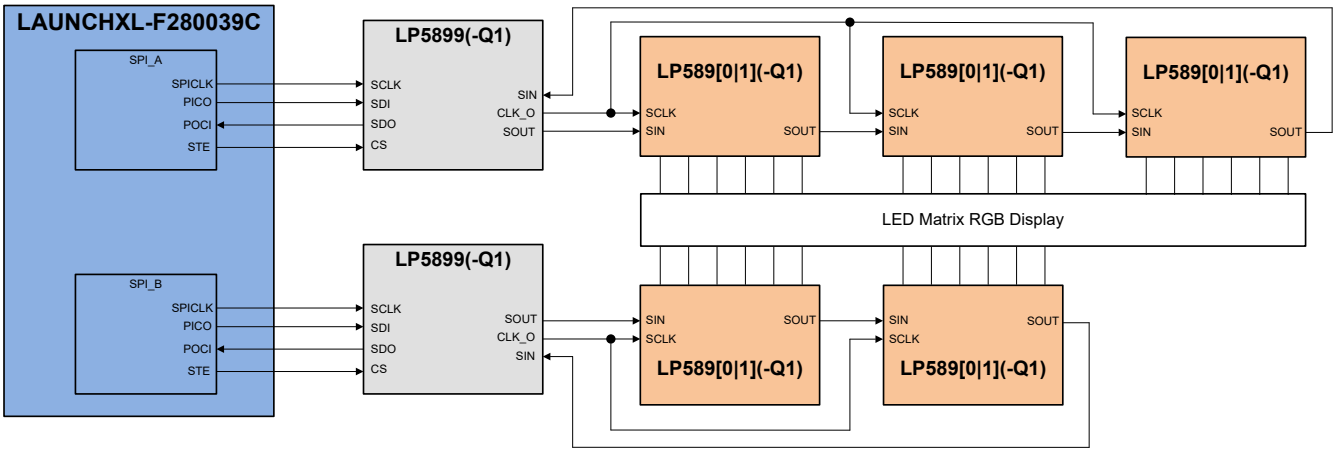

图 3-3. 使用增强连接 IC 和 2 个 CCSI 链的示例系统

示例代码最多支持 2 个 CCSI 菊花链。图 3-2 和图 3-3 中的示例用于展示支持 2 个 CCSI 菊花链的情况。第一个图显示了不使用增强连接 IC 的示例，第二个图显示了使用增强连接 IC 的示例。对于这两个示例，实际使用的链数由文件 `system_info.h` 中的宏 *CCSI_BUS_NUM* 定义。

```
// Total CCSI buses supported
#define CCSI_BUS_NUM            2
```

每个链可具有不同数量的级联器件。因此，除了宏 *CASCADED_UNITS_CCSI1* 外，文件 `system_info.h` 中还有宏 *CASCADED_UNITS_CCSI2*。在这些示例中，CCSI 链 1 有 3 个级联器件，CCSI 链 2 有 2 个级联器件。

```
#define CASCADED_UNITS_CCSI1    3
#define CASCADED_UNITS_CCSI2    2
```

3.4 诊断

示例代码提供了一个 API 来检测哪些 LED 驱动器具有 LED 开路 (LOD) 故障以及哪些 LED 驱动器具有 LED 短路 (LSD) 故障。此外，当示例代码与 LP5899DYYEVM 配对使用时，将读取 LP5899 增强型连接器件检测到的故障。TLC698x_LP589x_APIs.h 文件中定义了该 API 的原型。

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

该 API 会更新 system_info.h 中定义的变量 *chip_status*。

```
struct ccsiBusStatus {
    uint16_t LSD;             // LED Short Detection
    uint16_t LOD;             // LED Open Detection
    volatile outChannel LOD_channels[TOTAL_SCAN_LINES];
    volatile outChannel LSD_channels[TOTAL_SCAN_LINES];
};

struct chipStatus {
    struct ccsiBusStatus busStatus[MAX_CASCADED_UNITS];
#ifdef _TLC6989_LP5899
    uint16_t ERR;                   // LP5899/TLC6989 Global error flag
    uint16_t POR;                   // Power-On-Reset flag
    uint16_t OSC;                   // Internal oscillator flag
    uint16_t OTP_CRC;               // OTP CRC flag
    uint16_t DEV_STATE;             // Device state
    uint16_t SPI_CRC;               // SPI CRC error
    uint16_t SPI_REG_WRITE;         // SPI register write error
    uint16_t SPI;                   // SPI error
    uint16_t SPI_CS;                // SPI Chip Select (CS) pin error
    uint16_t SPI_TIMEOUT;           // SPI reset timeout error
    uint16_t SRST;                  // Softreset error
    uint16_t DRDY_STATUS;           // DRDY pin status
    uint16_t RXFF;                  // Receive FIFO error
    uint16_t RXFFSED;               // Receive FIFO single error detection
    uint16_t RXFFUVF;               // Receive FIFO underflow
    uint16_t RXFFOVF;               // Receive FIFO overflow
    uint16_t TXFF;                  // Transmit FIFO error
    uint16_t TXFFSED;               // Transmit FIFO single error detection
    uint16_t TXFFUVF;               // Transmit FIFO underflow
    uint16_t TXFFOVF;               // Transmit FIFO overflow
    uint16_t CCSI;                  // CCSI interface error
    uint16_t CCSI_SIN;              // CCSI missing toggling on SIN error
    uint16_t CCSI_CRC;              // CCSI data CRC error
    uint16_t CCSI_CHECK_BIT;        // CCSI check bit error
    uint16_t CCSI_CMD_QUEUE_OVF;    // CCSI command queue overflow
#endif
};

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

请注意，在示例代码中，诊断仅在简单测试模式期间执行，而不是在动画模式期间执行。

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 3-4 中描述了一个没有任何错误的 LP5891Q1EVM 示例。变量 *chip_status* 的第一个索引是 CCSI 链索引。在 EVM 上，只使用了 1 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。

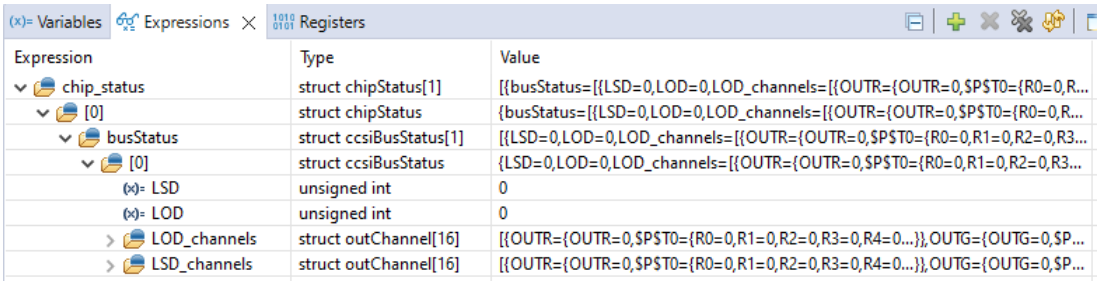

图 3-4. 观察表达式 chip_status（无错误）的示例

图 3-5 显示了 *chip_status* 变量的扩展视图，其中展示了链中每个 LED 驱动器器件的 LSD 和 LOD。此外，对于 LSD 和 LOD 故障，可以找到具有故障的实际线路和输出通道。LP5891Q1EVM 使用 16 条扫描线。因此，LOD_channels 阵列的索引在 0 至 15 之间。

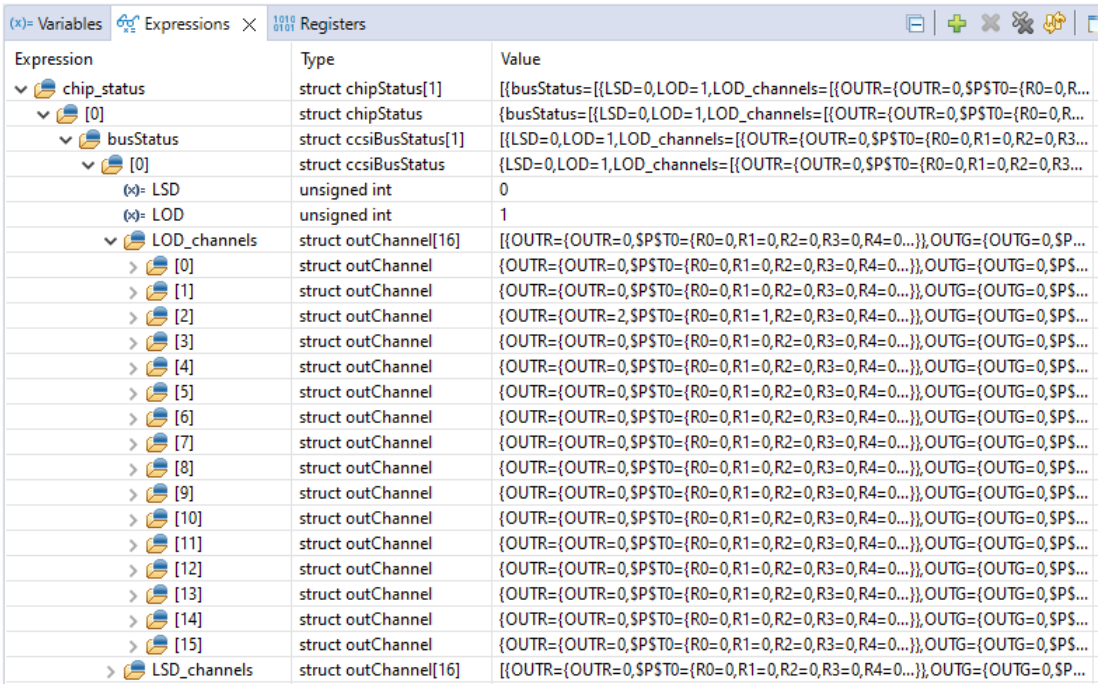

图 3-5. 显示具有 LOD 的 chip_status 表达式的扩展扫描线的示例

LOD 故障的示例如图 3-6 所示。在此示例中，芯片索引 0 具有 LOD 故障。当 LOD_channels 阵列被扩展后，会显示故障发生在索引 2 对应的线上。该索引扩展后，会显示故障发生在引脚 R1 上。

(x)= Variables | Expressions | Registers

Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{LSD=0,LOD=1,LOD_channels=[{OUTR={OUTR=0,PT0={R0=0,R...
[0]	struct chipStatus	{busStatus=[{LSD=0,LOD=1,LOD_channels=[{OUTR={OUTR=0,PT0={R0=0,R...
busStatus	struct ccsiBusStatus[1]	[{LSD=0,LOD=1,LOD_channels=[{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3...
[0]	struct ccsiBusStatus	{LSD=0,LOD=1,LOD_channels=[{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3...
LSD	unsigned int	0
LOD	unsigned int	1
LOD_channels	struct outChannel[16]	[{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3=0,R4=0...}},OUTG={OUTG=0,$P...
[0]	struct outChannel	{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3=0,R4=0...}},OUTG={OUTG=0,P...
[1]	struct outChannel	{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3=0,R4=0...}},OUTG={OUTG=0,P...
[2]	struct outChannel	{OUTR={OUTR=2,PT0={R0=0,R1=1,R2=0,R3=0,R4=0...}},OUTG={OUTG=0,P...
OUTR	union <unnamed>	{OUTR=2,PT0={R0=0,R1=1,R2=0,R3=0,R4=0...}}
OUTR	unsigned int	2
PT0	struct <unnamed>	{R0=0,R1=1,R2=0,R3=0,R4=0...}
R0	unsigned int : 1	0
R1	unsigned int : 1	1
R2	unsigned int : 1	0
R3	unsigned int : 1	0
R4	unsigned int : 1	0
R5	unsigned int : 1	0
R6	unsigned int : 1	0
R7	unsigned int : 1	0
R8	unsigned int : 1	0
R9	unsigned int : 1	0
R10	unsigned int : 1	0
R11	unsigned int : 1	0
R12	unsigned int : 1	0
R13	unsigned int : 1	0
R14	unsigned int : 1	0
R15	unsigned int : 1	0
OUTG	union <unnamed>	{OUTG=0,PT1={G0=0,G1=0,G2=0,G3=0,G4=0...}}
OUTB	union <unnamed>	{OUTB=0,PT2={B0=0,B1=0,B2=0,B3=0,B4=0...}}
[3]	struct outChannel	{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3=0,R4=0...}},OUTG={OUTG=0,P...

图 3-6. 显示具有 LOD 的 chip_status 表达式的扩展通道的示例

当 LP5899DYYEVM 与 LP589x(-Q1) 配对使用时，*chip_status* 变量还会显示增强连接 IC 的错误标志。图 3-7 中描述了一个没有任何故障的示例，其中为每个 CCSI 链（*chip_status* 的第一个索引）列出了错误标志。请注意，DRDY_STATUS 为 1，因为当 LP5899 具有已准备好被读取的数据时，该值变为零。

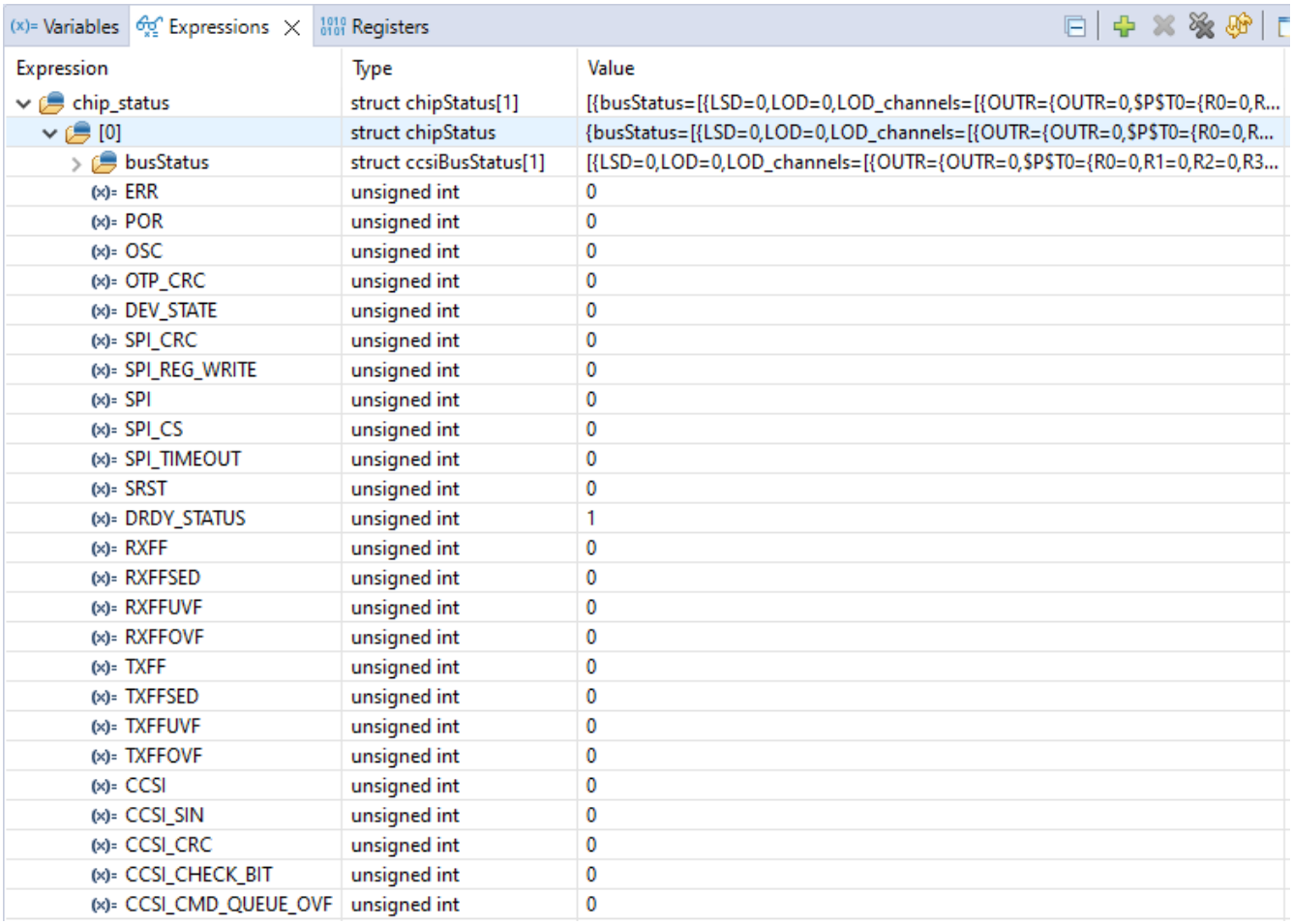

图 3-7. 与 LP5899 配对使用时未出现错误的 chip_status 示例

CCSI 故障的示例如图 3-8 所示。在本示例中，SIN 引脚卡在高电平，而 CCSI 传输继续，这也会导致 CCSI 命令队列溢出。

Variables | Expressions | Registers

Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{LSD=0,LOD=0,LOD_channels=[{OUTR={OUTR=0,PT0={R0=0,R...
[0]	struct chipStatus	{busStatus=[{LSD=0,LOD=0,LOD_channels=[{OUTR={OUTR=0,PT0={R0=0,R...
busStatus	struct ccsiBusStatus[1]	[{LSD=0,LOD=0,LOD_channels=[{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3...
ERR	unsigned int	1
POR	unsigned int	0
OSC	unsigned int	0
OTP_CRC	unsigned int	0
DEV_STATE	unsigned int	0
SPI_CRC	unsigned int	0
SPI_REG_WRITE	unsigned int	0
SPI	unsigned int	0
SPI_CS	unsigned int	0
SPI_TIMEOUT	unsigned int	0
SRST	unsigned int	0
DRDY_STATUS	unsigned int	1
RXFF	unsigned int	0
RXFFSED	unsigned int	0
RXFFUVF	unsigned int	0
RXFFOVF	unsigned int	0
TXFF	unsigned int	0
TXFFSED	unsigned int	0
TXFFUVF	unsigned int	0
TXFFOVF	unsigned int	0
CCSI	unsigned int	1
CCSI_SIN	unsigned int	1
CCSI_CRC	unsigned int	0
CCSI_CHECK_BIT	unsigned int	0
CCSI_CMD_QUEUE_OVF	unsigned int	1

图 3-8. 与 LP5899 配对使用时具有 CCSI 故障的 Expression chip_status 示例

3.5 演示

本节介绍了 LED 演示的示例。图 3-9 图示为运行演示的 LP5891Q1EVM 。

图 3-10 图示为运行演示的 TLC6984EVM。

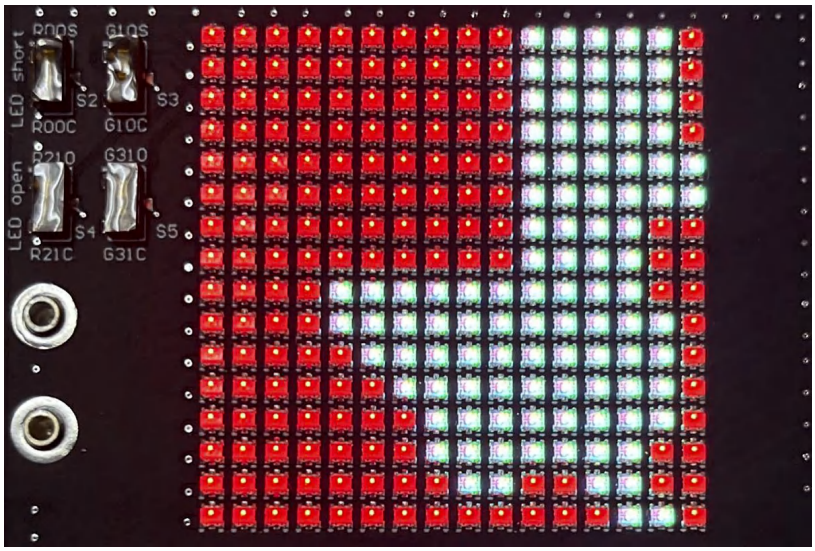

图 3-9. LP5891Q1EVM 演示示例

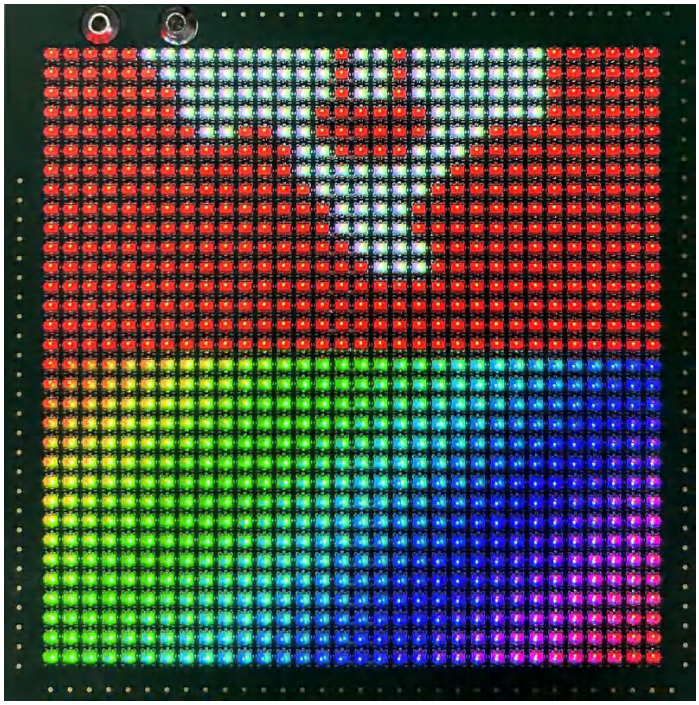

图 3-10. TLC6984EVM 演示示例

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

Changes from Revision * (May 2023) to Revision A (September 2024)	Page
• 通篇更新了整个 LP589x/TLC698x 器件系列	1
• 更新了节 2。	3
• 更新了节 3.2。	5
• 更新了节 3.3。	7
• 更新了节 3.4。	10

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