User's Guide

TLC6962x/TLC6963x-Q1 Sample Code

ABSTRACT

This document describes the preparation and usage of the sample code for TLC6962x/TLC6963x-Q1 device family when paired with a LAUNCHXL-F280039C. Following the instructions provided for setup, the installed code lights up the LEDs on the EVM. The TLC696[2|3][3|6|9] devices can be paired with the TLC69698-Q1 to use a standard SPI or UART module of the MCU to communicate to the LED drivers. The TLC696[2|3][1|2|4|5|7|8] devices can be paired with the TLC69697-Q1 to use a standard UART module of the MCU to communicate to the LED drivers providing capability for communication over multiple boards using a physical CAN layer.

Table of Contents

List of Figures

List of Tables

Trademarks

LaunchPad™ is a trademark of Texas Instruments.
Code Composer Studio™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

The sample code showcases the ability to light up the LEDs on the TLC69628Q1EVM, TLC69622Q1EVM, TLC69625Q1EVM, and TLC69698Q1EVM. Each EVM has a dedicated sample code. However, the only difference is in the file led_driver.h to select the used LED driver IC. This helps the user to be able to light up the EVM without any modication to the sample code.

There are two modes in the code: animation and simple test. The animation mode is selected by default. Section 3.3 describes how to change between the modes. In the animation mode, one frame is used to scroll left, right, up, and down and to fade in and fade out according to a predefined sequence. For TLC69628Q1EVM, TLC69622Q1EVM, and TLC69625Q1EVM the frame is a Texas Instruments logo of 99x6 pixels. For TLC69698Q1EVM the frame is a Texas Instruments logo of 99x8 pixels. This means that only part of the full frame is shown on the LED display of the EVM. Examples of this can be seen in Section 3.6. The method how to generate the frame in the sample code is outside the scope of this document.

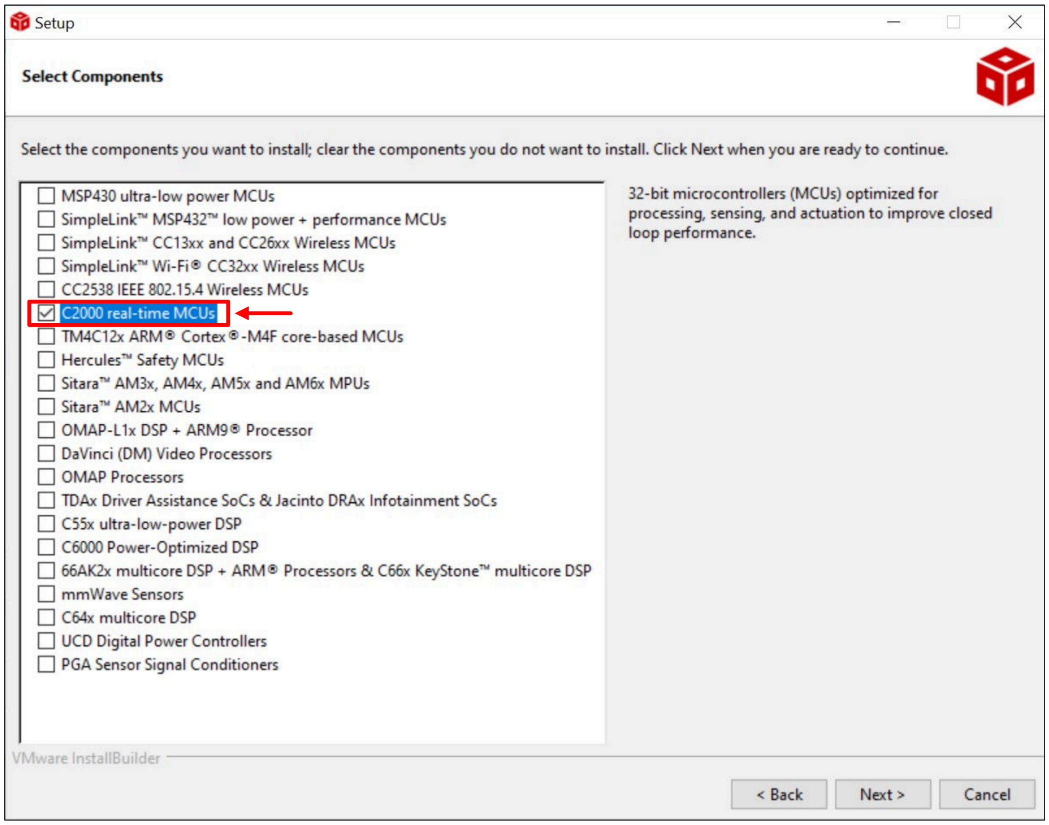
In the simple test mode, the user can use some predefined APIs to light up the LED board, perform diagnostics, or build custom interface commands. The sample code comes with turning on all LEDs using a 50% duty cycle. In addition, diagnostics is executed to determine any LED, LED driver faults, and when enabled universal plug-in augmented connectivity (UPAC) device faults. For more information about diagnostics see Section 3.5. The predefined APIs automatically adjust to the specified system. More detail about the system specification can be found in Section 3.3 and Section 3.4.

The TLC69698Q1EVM pairs the TLC69698-Q1 UPAC device with one TLC69629-Q1 LED driver and one TLC69623-Q1 LED driver. The sample code for TLC69698Q1EVM has already enabled the pairing and uses the LAUNCHXL-F280039C's SPI to communicate to the TLC69698-Q1. To change to UART, follow the jumper settings described in the hardware User's Guide. Software changes are described in Section 3.3. Using a UPAC device enables additional diagnostics which is described in more detail in Section 3.5.

2 Software Setup

To set up the software for the TMS320F280039C LaunchPad™, follow these steps (demonstrated in a computer with Windows 10 OS):

1. Download and install Code Composer Studio™ (CCS).
   a. Download Code Composer Studio integrated development environment (IDE) (Version 12.7.0).
   b. Follow the installation instructions to install Code Composer Studio. During the installation process, if you choose the "Setup type" to be "Custom Installation", make sure that you select "C2000 real-time MCUs" in "Select Components", as is marked with red box in Figure 2-1.



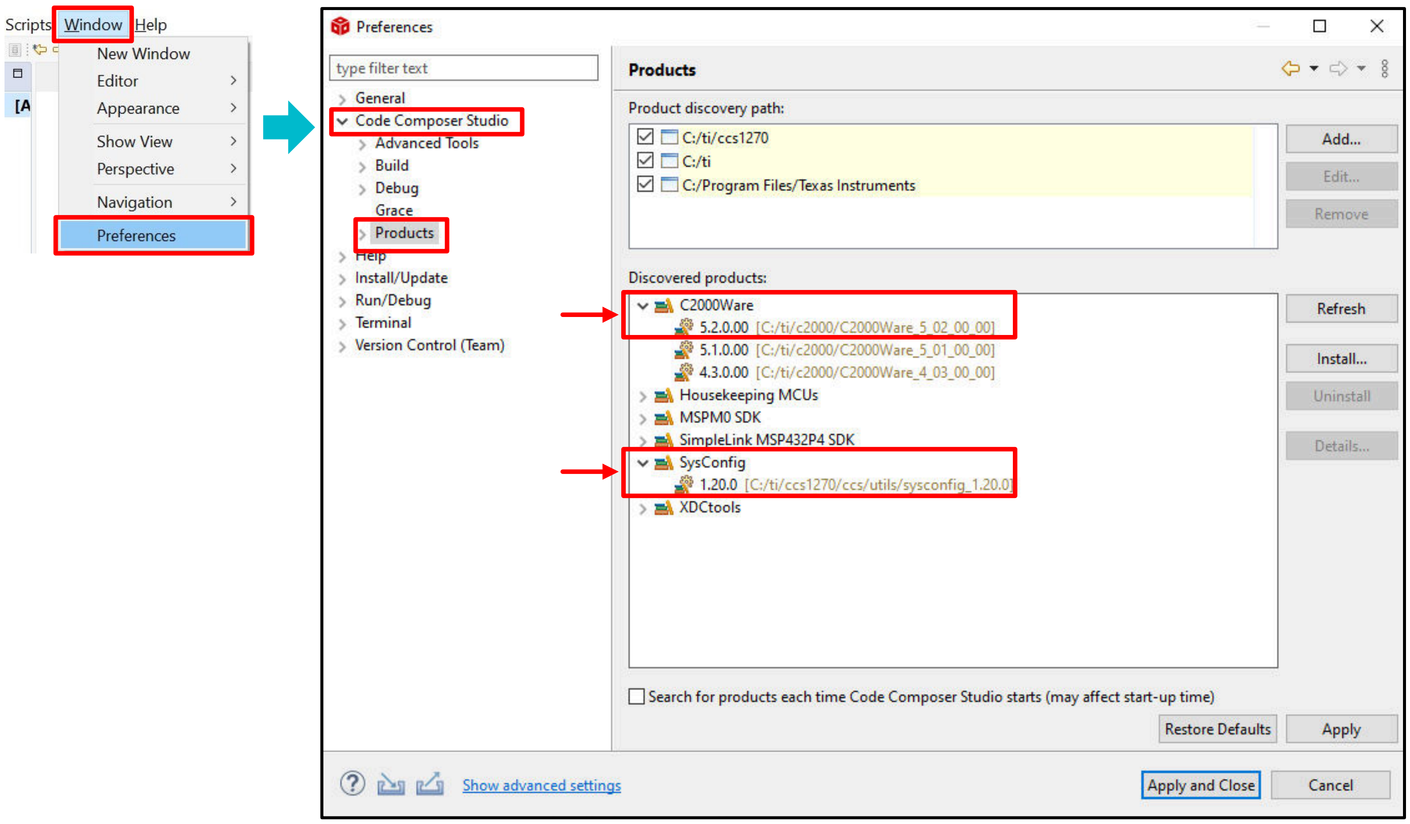
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

- [ ] MSP430 ultra-low power MCUs
- [ ] SimpleLink™ MSP432™ low power + performance MCUs
- [ ] SimpleLink™ CC13xx and CC26xx Wireless MCUs
- [ ] SimpleLink™ Wi-Fi® CC32xx Wireless MCUs
- [ ] CC2538 IEEE 802.15.4 Wireless MCUs
- [✓] C2000 real-time MCUs
- [ ] TM4C12x ARM® Cortex®-M4F core-based MCUs
- [ ] Hercules™ Safety MCUs
- [ ] Sitara™ AM3x, AM4x, AM5x and AM6x MPUs
- [ ] Sitara™ AM2x MCUs
- [ ] OMAP-L1x DSP + ARM9® Processor
- [ ] DaVinci (DM) Video Processors
- [ ] OMAP Processors
- [ ] TDAx Driver Assistance SoCs & Jacinto DRAx Infotainment SoCs
- [ ] C55x ultra-low-power DSP
- [ ] C6000 Power-Optimized DSP
- [ ] 66AK2x multicore DSP + ARM® Processors & C66x KeyStone™ multicore DSP
- [ ] mmWave Sensors
- [ ] C64x multicore DSP
- [ ] UCD Digital Power Controllers
- [ ] PGA Sensor Signal Conditioners

32-bit microcontrollers (MCUs) optimized for processing, sensing, and actuation to improve closed loop performance.

Figure 2-1. Installation Process of Code Composer Studio

2. Download and install C2000Ware (Version 5.02.00.00).
   a. To verify the installation is correct, open the Code Composer Studio software. Click "Windows" from the top menu bar. Then click "Preferences" from the drop-down list. The "Preferences" window appears. From the left side bar, select "Code Composer Studio" -> "Products".
   b. Make sure that there is "C2000Ware 5.2.0.00" and "SysConfig" (SysConfig is installed automatically when you install C2000Ware) under the "Discovered products", as is marked in Figure 2-2. If there is no "C2000Ware 5.2.0.00" or "SysConfig" appearing, click "Refresh" on the right side of the "Preference" window. If still unavailable, check whether the installation is correct. A standalone desktop version of SysConfig can be found at SYSCONFIG System configuration tool.



Product discovery path:

- [✓] C:/ti/ccs1270
- [✓] C:/ti
- [✓] C:/Program Files/Texas Instruments

Discovered products:

- C2000Ware
  - 5.2.0.00 [C:/ti/c2000/C2000Ware_5_02_00_00]
  - 5.1.0.00 [C:/ti/c2000/C2000Ware_5_01_00_00]
  - 4.3.0.00 [C:/ti/c2000/C2000Ware_4_03_00_00]
- Housekeeping MCUs
- MSPM0 SDK
- SimpleLink MSP432P4 SDK
- SysConfig
  - 1.20.0 [C:/ti/ccs1270/ccs/utils/sysconfig_1.20.0]
- XDCtools

[ ] Search for products each time Code Composer Studio starts (may affect start-up time)

Figure 2-2. Verification of C2000Ware 5.02.0.00 Installation

3. Download and import sample code.
   a. Importing the Code Composer Studio (CCS) project according to the process provided in the link: Importing a CCS Project.
4. Before loading and running the program, set switch S2 on LAUNCHXL-F280039C for both SEL1 and SEL2 to the 1 position.
5. Load the program according to the process provided in the link: Building and Running Your Project.

3 Sample Code Structure

3.1 Design Parameters

The LED matrix display design parameters used for the EVM are listed in Table 3-1.

Table 3-1. Design Parameters EVM

Design Parameter	TLC69628Q1EVM	TLC69698Q1EVM	TLC69622Q1EVM	TLC69625Q1EVM
Display module size	16 x 6 white LEDs	7 x 8 white LEDs	4 x 2 white LEDs	4 x 6 white LEDs
Frame rate	60Hz	60Hz	60Hz	60Hz
PWM resolution	16 bits	16 bits	16 bits	16 bits
Cascaded devices	2	2	1	1
CLK_I frequency	3MHz	3MHz	3MHz	3MHz

3.2 Flow Diagram

Figure 3-1 depicts the high level flow in the sample code. During the Setup MCU some inputs are coming from the file system_info.h. Depending on the *IF_SEL* macro, the MCU setups the SPI module or the UART module. When UART is selected, the default data rate 515625bps is chosen. When SPI is selected, the clock frequency for the SPI communication uses *SPI_FREQ_IN_HZ* macro. The system_info.h file is described in more in detail in Section 3.3 and Section 3.4.

When the LED driver registers are going to be written, most data comes from the LED_Reg_settings.h file. However, the number of cascaded devices (field CHIP_NUM in register DEVSET) is coming from the file system_info.h.

The flow diagram also shows the files frames.c and frames.h, which contain the frame used during the animation mode. The method how to generate the frame in the sample code is outside the scope of this document.

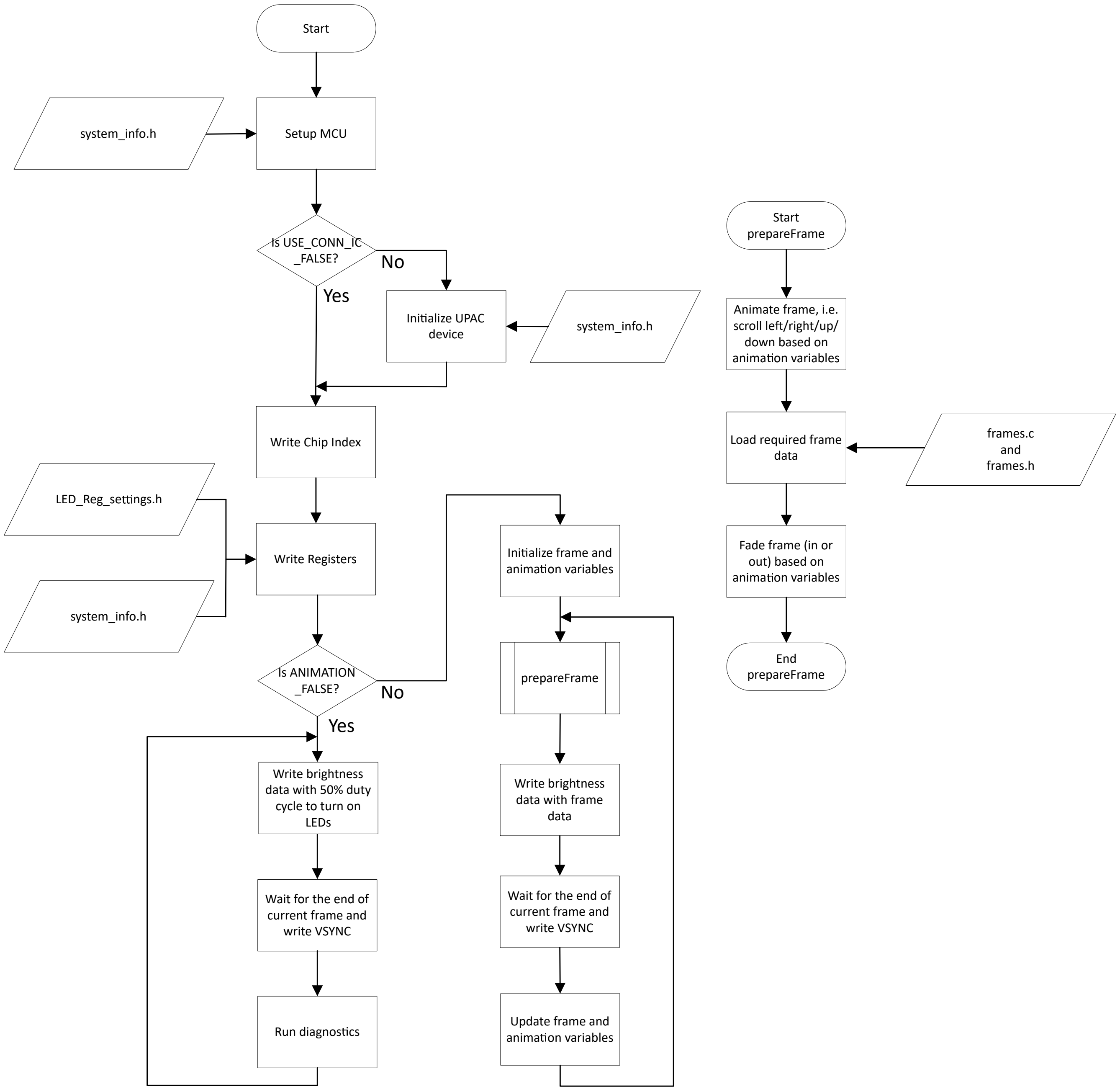

Start
system_info.h
Setup MCU
Is USE_CONN_IC _FALSE?
No
Yes
Initialize UPAC device
system_info.h
Write Chip Index
LED_Reg_settings.h
Write Registers
system_info.h
Is ANIMATION _FALSE?
No
Yes
Write brightness data with 50% duty cycle to turn on LEDs
Wait for the end of current frame and write VSYNC
Run diagnostics
Initialize frame and animation variables
prepareFrame
Write brightness data with frame data
Wait for the end of current frame and write VSYNC
Update frame and animation variables
Start prepareFrame
Animate frame, i.e. scroll left/right/up/down based on animation variables
Load required frame data
frames.c and frames.h
Fade frame (in or out) based on animation variables
End prepareFrame

Figure 3-1. Sample Code Flow Diagram

3.3 Basic System Setup

This section describes how the sample code setups different parameters to identify how the system is built.

The first part is the actual used LED driver IC. Within the led_driver.h file, the selection for the used LED driver IC is made.

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

The code supports the below products:

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

Note that for the "Q1" devices, this is not added and only the base product name is used.

A summary about macros and variables that impact the system setup and macros and variables location is listed in Table 3-2.

Table 3-2. Summary of Macro and Variable Names Per File

Filename	Macro/Variable name	Description
system_info.h	*SPI_FREQ_IN_HZ*	CLK_I frequency (LED drivers) or SCLK_RX frequency (TLC69698-Q1) when *IF_SEL* is defined as *SPI*.
	USE_CONN_IC	Selection if UPAC device TLC69698-Q1/TLC69697-Q1 is used.
	RUN_MODE	Selection between different code run modes. Supported options are *ANIMATION* and *SIMPLE_TEST*.
	IF_SEL	Selection of MCU interface used to communicate.
	TOTAL_SCAN_LINES	Number of scan lines. For TLC6962x/TLC6963x-Q1 family this is always 1.
	BUS_NUM	Number of daisy chain buses.
	CASCADED_UNITS_BUS1	Device count in daisy chain bus 1.
	CASCADED_UNITS_BUS2	Device count in daisy chain bus 2.
	CASCADED_UNITS_BUS3	Device count in daisy chain bus 3. Only supported for UART with UPAC device.
	CASCADED_UNITS_BUS4	Device count in daisy chain bus 4. Only supported for UART with UPAC device.
	SCAN_FET_CTRL_NUM	Number of SCAN FET controllers in one daisy chain. For TLC6962x/TLC6963x-Q1 family this is always 0.
system_info.c	*FRAME_RATE*	Interval of VSYNC commands.
	dev_addr	Device addresses of connectivity devices used on the UART bus.

Within the file system_info.c the frame rate and UART device addresses of the UPAC devices are specified. The frame period is specified in Hz (frames per second). The minimum supported frame rate is 1Hz.

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

Variable *dev_addr* only needs to be set when *USE_CONN_IC* is set to _TRUE and *IF_SEL* is set to *UART* or *UART_CAN*. The variable specifies the addresses of the UPAC devices on the UART bus.

File system_info.h includes several system definitions.

```
// 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
```

Macro *SPI_FREQ_IN_HZ* defines at what data rate the SPI is running, such that, the clock frequency of pin CLK_I for the LED drivers or pin SCLK_RX of TLC69698-Q1. This frequency is an integer divider from the system frequency of the MCU. Therefore, the actual SPI frequency sometimes differ from the desired specified frequency defined by *SPI_FREQ_IN_HZ*.

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

Macro *USE_CONN_IC* defines if the UPAC device is used. When LED driver is selected as TLC696[2|3][3|6|9], this is automatically set to _TRUE.

Macro *RUN_MODE* defines the run mode of the code. The supported run modes are animation and simple test mode.

Options for macro *IF_SEL* are *SPI*, *UART*, and *UART_CAN*. For *UART* and *UART_CAN* the selection of UPAC device *USE_CONN_IC* has to be set to _TRUE. The difference between *UART* and *UART_CAN* is that *UART* uses a direct UART connection between MCU and UPAC device while *UART_CAN* uses the CAN physical layer in between the MCU and UPAC device.

The following code block shows macros that impact the register settings.

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

Macro *TOTAL_SCAN_LINES* defines the number of scan lines used in the system. For TLC6962x/TLC6963x-Q1 family this is always 1.

Macro *CASCADED_UNITS_BUS1* defines the number of cascaded devices in the system and directly impacts the field CHIP_NUM is register DEVSET. For the TLC69628Q1EVM and TLC69698Q1EVM there are two devices cascaded while for TLC69622Q1EVM and TLC69625Q1EVM this is only one device.

3.4 Advanced System Setup

To describe the advanced system setup, example are used that are depicted in Figure 3-2, Figure 3-3, and Figure 3-4.

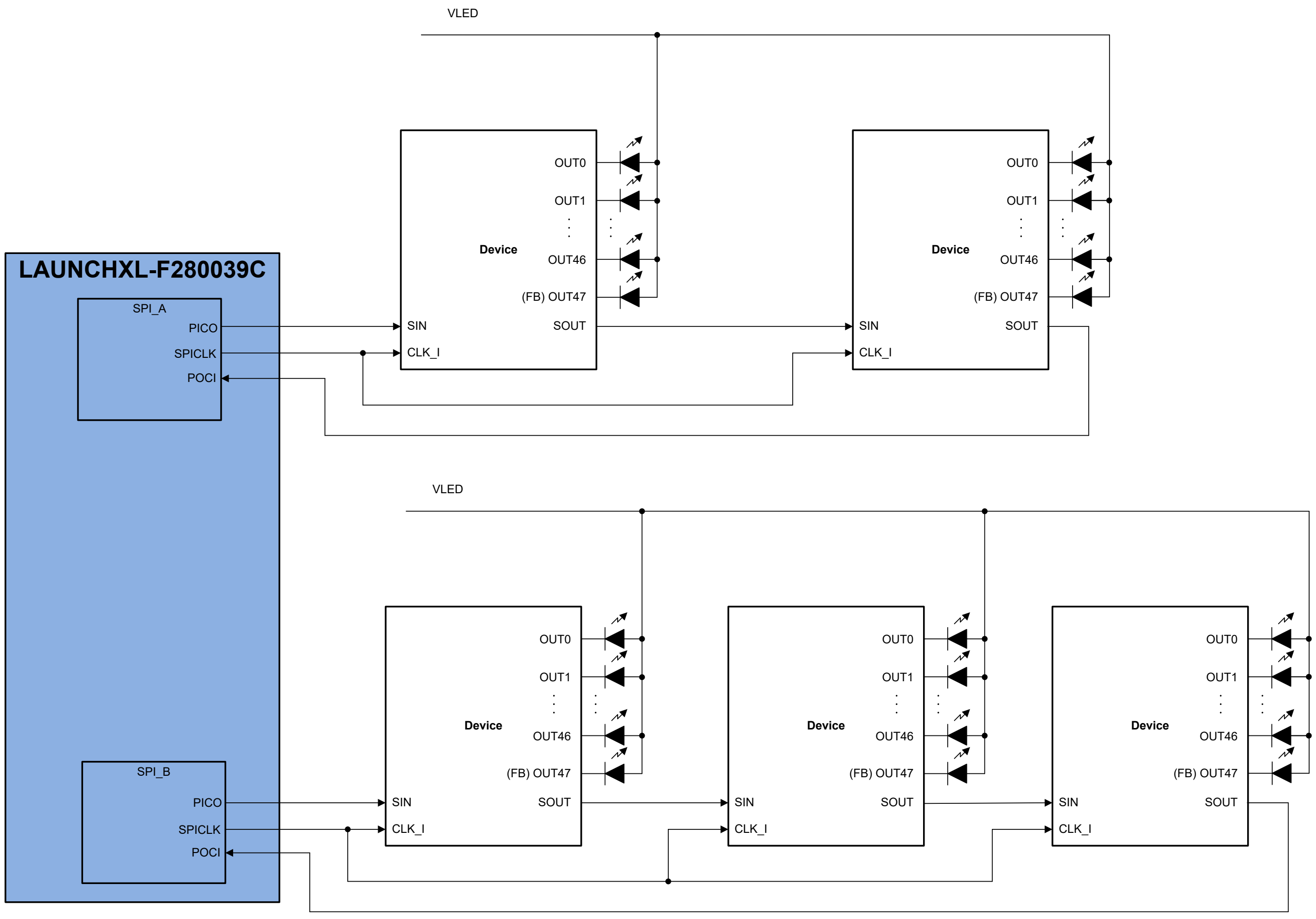


Figure 3-2. Advanced System Example With 2 Daisy Chains with direct SPI connection

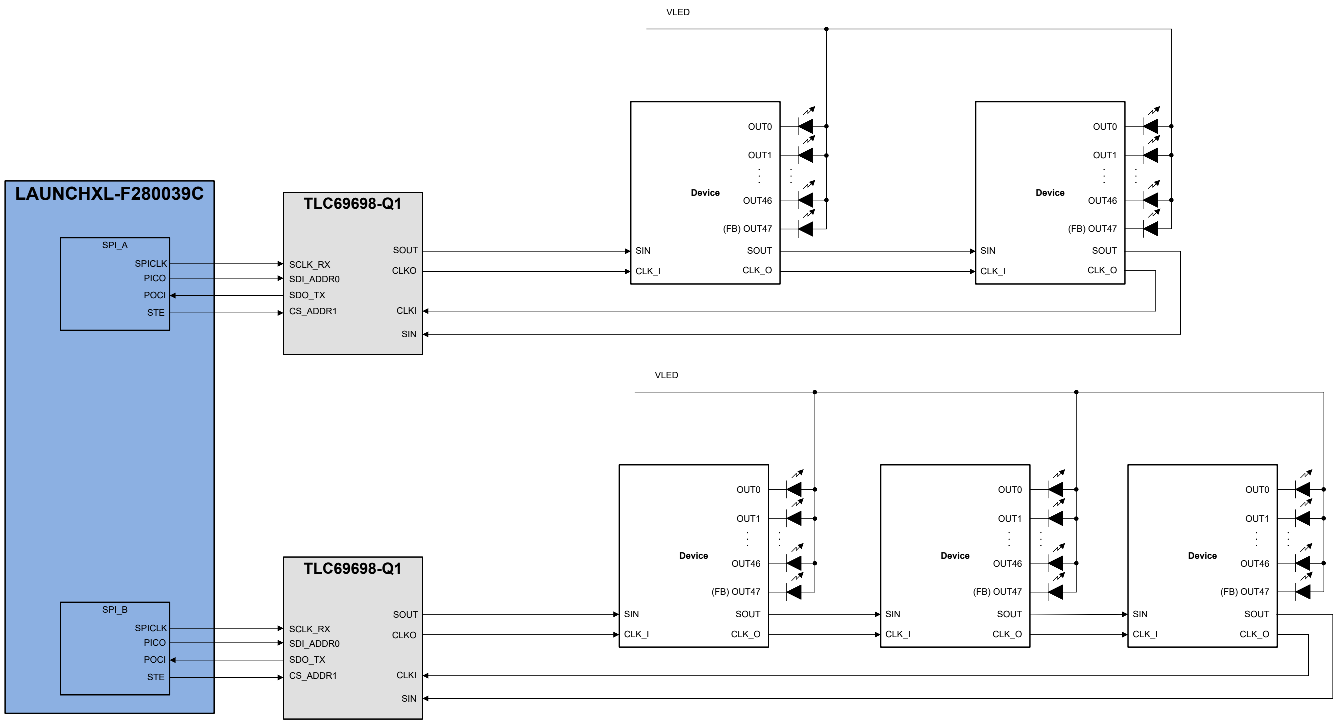


Figure 3-3. Advanced System Example With 2 Daisy Chains with SPI connection to UPAC device

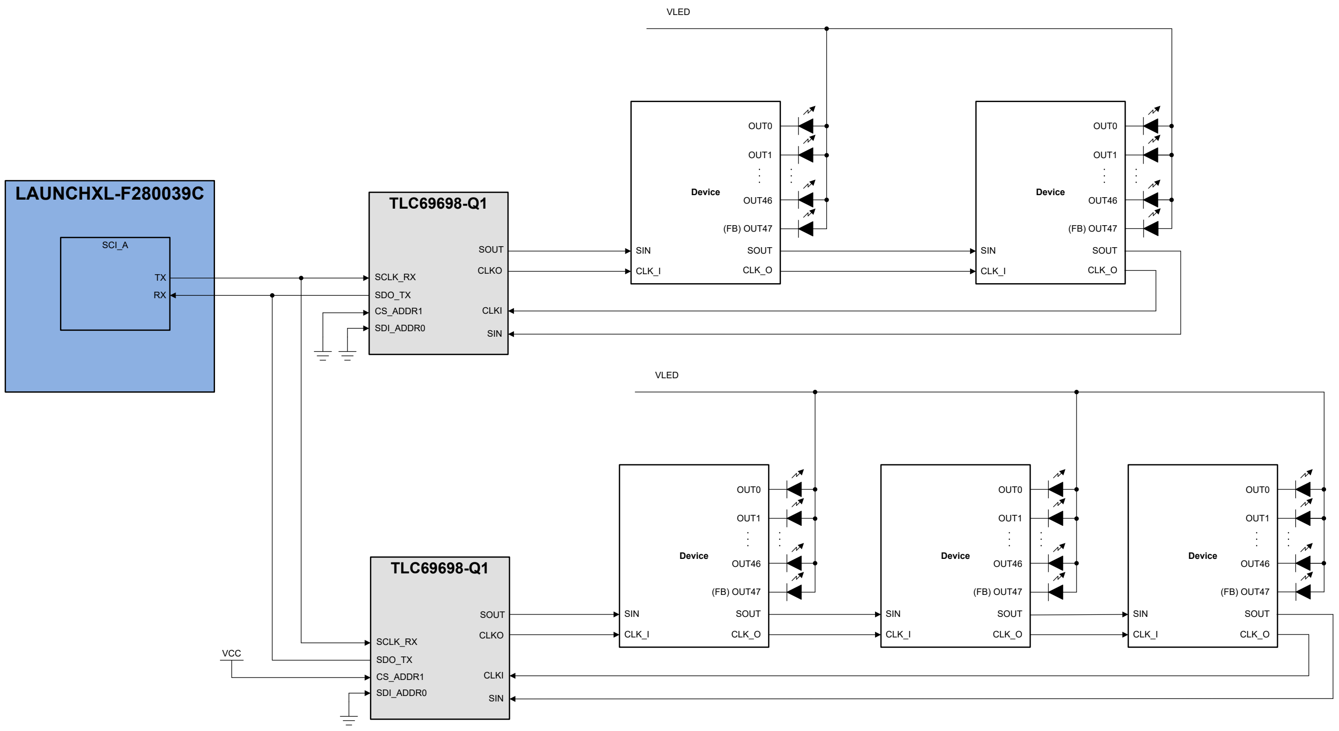


Figure 3-4. Advanced System Example With 2 Daisy Chains with UART connection to UPAC device

The sample code supports up to two daisy chains when using SPI. For UART, the limit is four daisy chains where the limit comes from the TLC69698-Q1/TLC69697-Q1. The number of actual used chains is defined by macro *BUS_NUM* in file system_info.h.

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

Each chain can have a different number of cascaded devices. Therefore, besides macro *CASCADED_UNITS_BUS1* which is described in Section 3.3, there is also macros *CASCADED_UNITS_BUS2*, *CASCADED_UNITS_BUS3* (only supported for UART), and *CASCADED_UNITS_BUS4* (only supported for UART) in file system_info.h. In the examples, chain 1 has two cascaded devices and chain 2 has three cascaded devices.

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

Note that for example depicted in Figure 3-4, *BUS_NUM* is for the number of daisy chains. There is still only one UART bus supported. For UART also *dev_addr* has to be set. In the example, one TLC69698-Q1 has address 0 and the other TLC69698-Q1 has address 2 (SDI_ADDR0 pulled low and CS_ADDR1 pulled high). The order of addresses has to match with the CASCADED_UNITS_BUSx definitions.

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

3.5 Diagnostics

The sample code provides an API to detect which LED driver devices have faults such as Thermal Shutdown (TSD), LED open (LOD), LED short (LSD), and Adjacent Pin Short (APS). For the LOD, LSD, LED_CHK, DRV_CHK, and APS, the failing channel is also detected. In addition, when macro *USE_CONN_IC* is set to _TRUE, the faults detected by the UPAC device are read. Within the TLC6962x_3x_APIs.h file, the prototype of the API is defined.

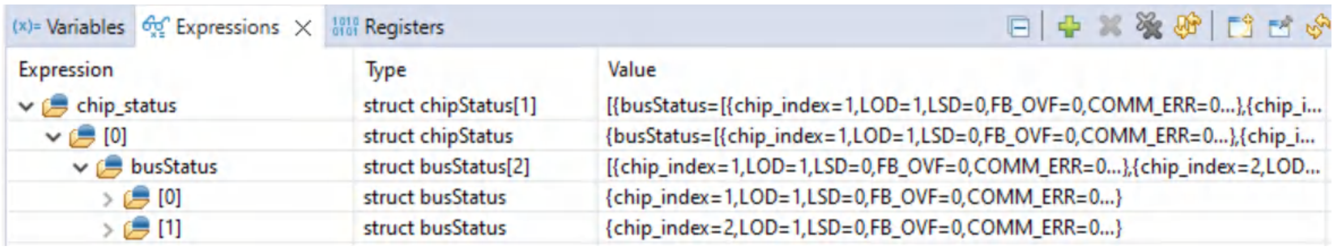
```
void LED_Update_Chip_Status(void);
```

This API updates the variable *chip_status* which is defined in file system_info.h.

```
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.
```

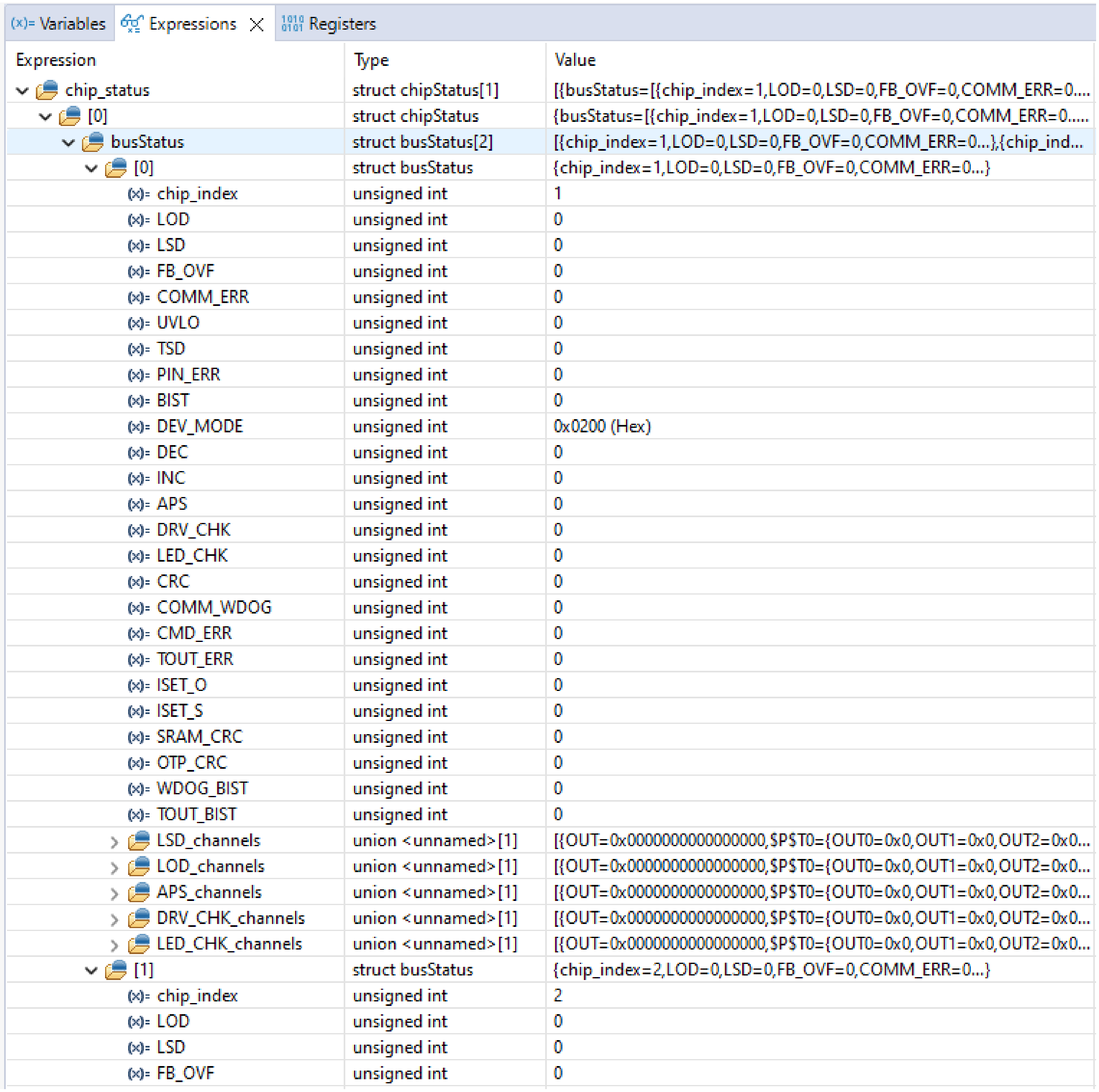
```
        uint16_t FLTSUM_DATAVALID;          // Fault summary data valid flag.
        uint16_t DEV_CNT_LOD_LSD;           // Device count that have LOD and/or LSD.
        uint16_t ZONE_CNT_LOD_LSD;          // Total zone count that have LOD and/or LSD.
        uint16_t SZ_CNT_LOD_LSD;          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

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

The variable *chip_status* can be watched in the Expressions view during the debug of the code by following the steps in Watching Variables, Expressions, and Registers. An example without any error is depicted in Figure 3-5. The first index of variable *chip_status* is the daisy chain index. On the EVMs there is only one chain. For the flags of the LED drivers, the busStatus variable is used which has an index for each LED driver in the chain. On the TLC69628Q1EVM and TLC69698Q1EVM there are 2 devices cascaded and, therefore, busStatus runs from 0 to 1. For TLC69622Q1EVM and TLC69625Q1EVM there is only 1 device cascaded and, therefore, busStatus only has index 0.



Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_i...
[0]	struct chipStatus	{busStatus=[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_i...
busStatus	struct busStatus[2]	[{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_index=2,LOD...
[0]	struct busStatus	{chip_index=1,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...}
[1]	struct busStatus	{chip_index=2,LOD=1,LSD=0,FB_OVF=0,COMM_ERR=0...}

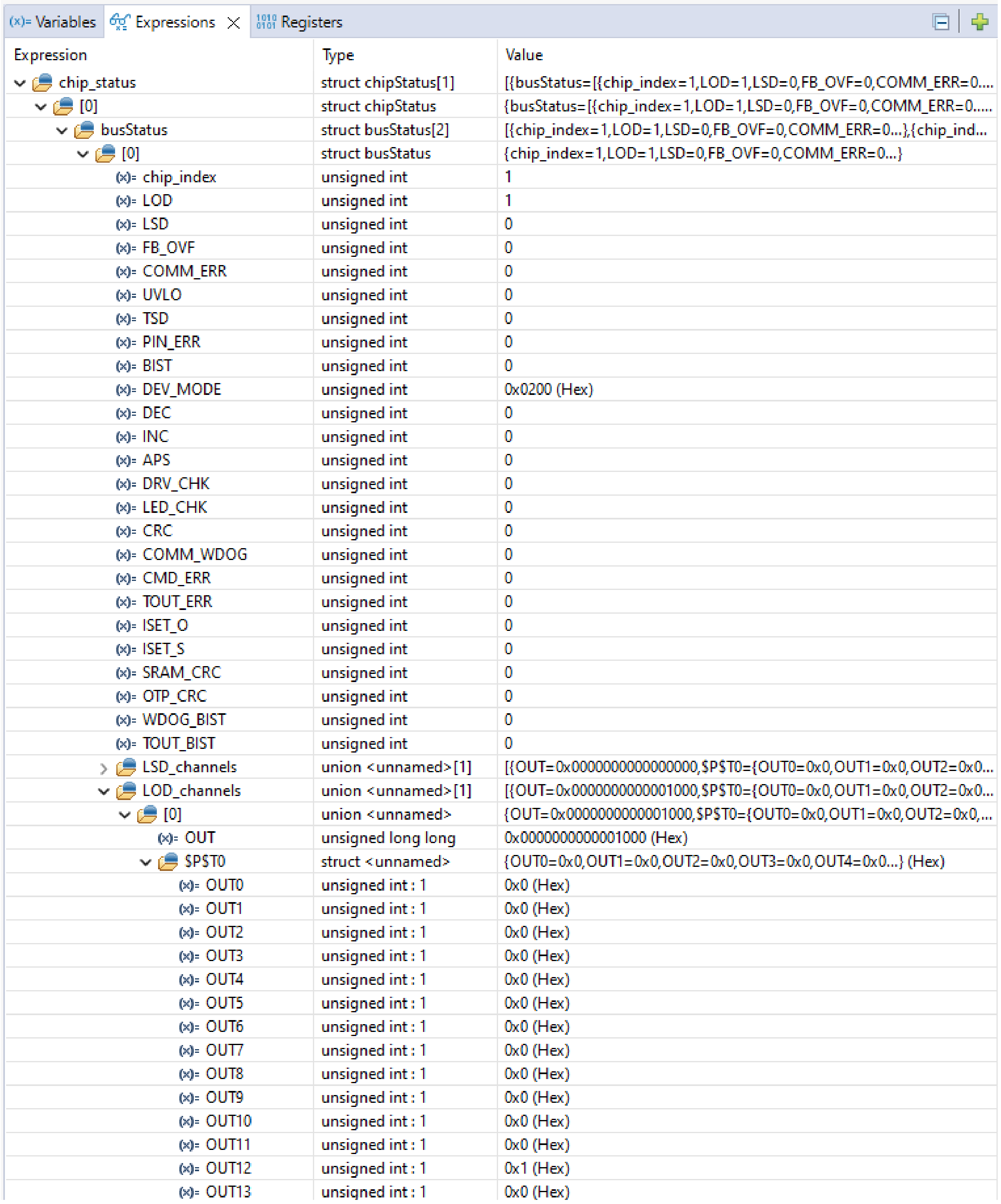
Figure 3-5. Example of Watching Expression chip_status Without Errors


(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

Figure 3-6. Expanded Example of Watching Expression chip_status Without Errors

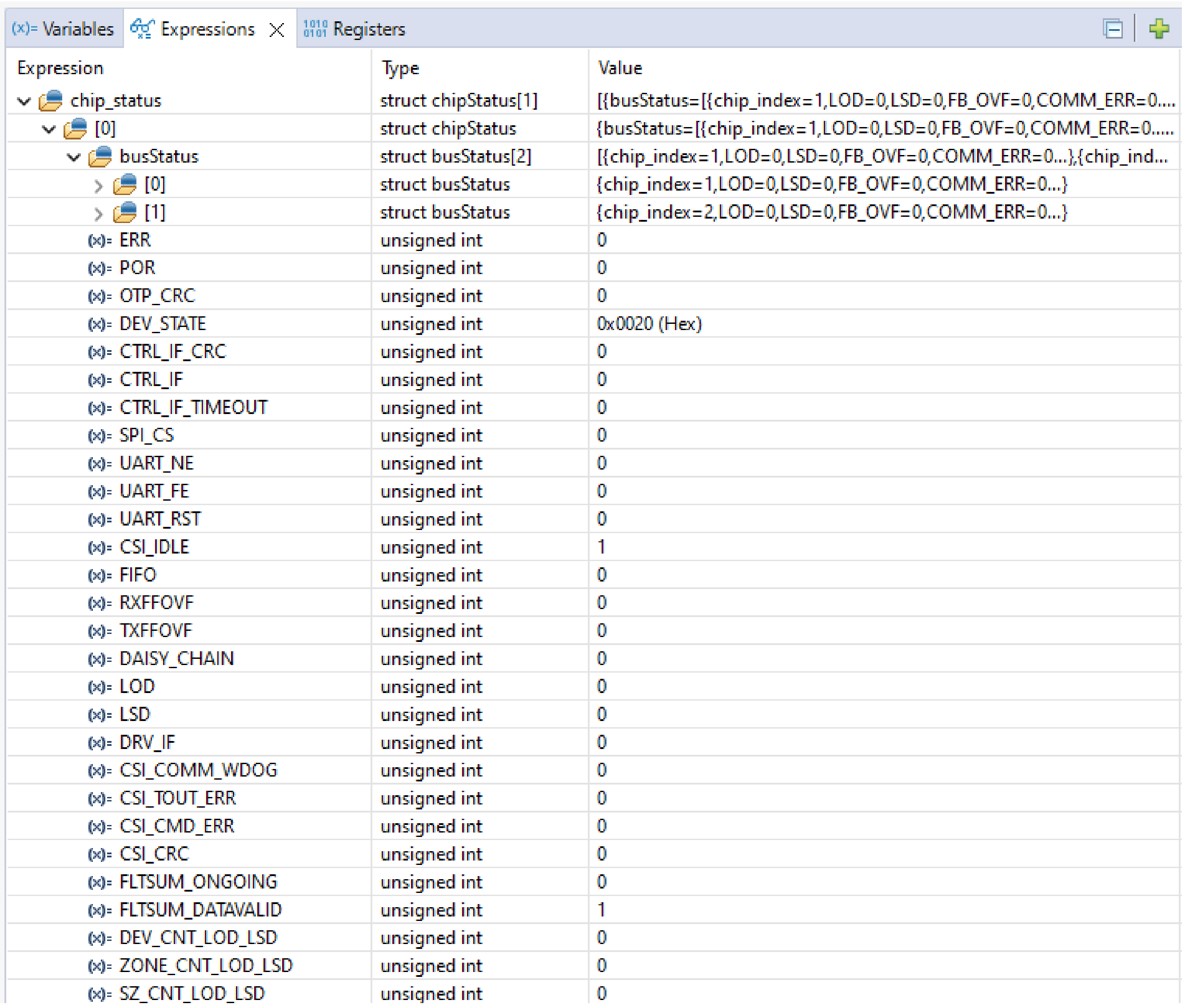
Figure 3-6 depicts an expanded view of the *chip_status* variable. For each of the two LED drivers in the daisy chain, the chip index and fault status bits are shown. In addition, for the LSD, LOD, LED_CHK, DRV_CHK, and APS faults the actual output channel that has the fault can be found. An example of LOD fault is depicted in Figure 3-7. In this example, chip index 1 (which is index 0 of busStatus) has an LOD fault. When array LOD_channels is expanded, the expansion shows that the fault occurs on pin OUT12. The LOD_channels is an array with only 1 index because for the TLC6962x/TLC6963x-Q1 devices there is only 1 scan line.



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)

Figure 3-7. Example of chip_status With LED Open (LOD) Fault

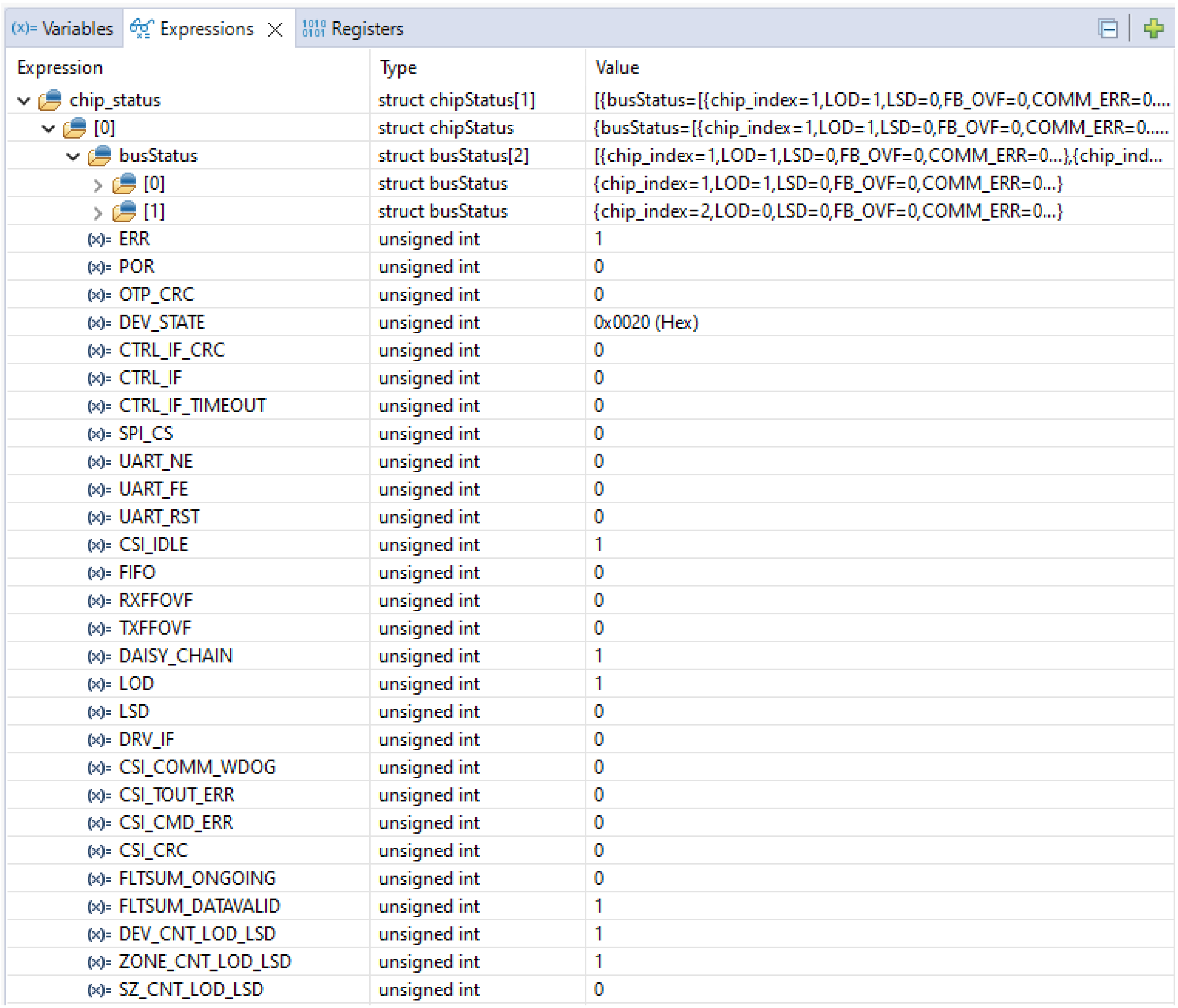
When the LED drivers are paired with the UPAC device, the *chip_status* variable also shows the error flags of the UPAC device. An example without any fault is depicted in Figure 3-8. For each daisy chain (first index of *chip_status*) the error flags are listed. Note that CSI_IDLE is 1, because during the diagnostics no messages is forwarded by the UPAC device. FLTSUM_DATAVALID is 1 because the diagnostics in the sample code happens after the first cycle of the Fault Summary of TLC69698-Q1.



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

Figure 3-8. Example of chip_status Without Errors When Paired With TLC69698-Q1

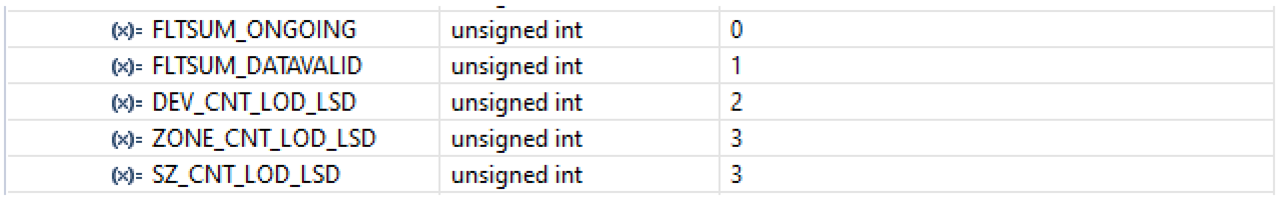
An example of an LOD fault when using the UPAC device is depicted in Figure 3-9. In this example both DAISY_CHAIN and LOD flags in TLC69698-Q1 are set. Moreover, the fault summary feature of the TLC69698-Q1 counted that there is 1 device with a fault (DEV_CNT_LOD_LSD) and that there is 1 LED zone with a fault (ZONE_CNT_LOD_LSD). The LED zone which is failing is not indentified as a so-called safety zone in the LED driver. Therefore, SZ_CNT_LOD_LSD is still 0.



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

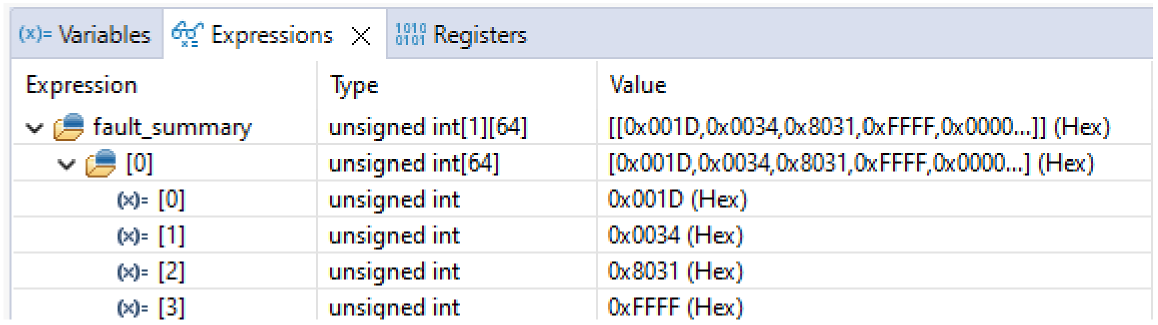
Figure 3-9. Example Showing Expression chip_status With LOD Fault When Paired With TLC69698-Q1

In the example depicted in Figure 3-10 the fault summary counted 3 LED faults in 2 devices where all 3 LED zones have been specified as so-called safety zones. When safety zones fail, the TLC69698-Q1 also records the location of the faults. Those locations can be found in variable *fault_summary* which is depicted in Figure 3-11. The entry after the last valid entry is filled with 0xFFFF to indicate the end of the array. That means that in the example index 3 of the array is filled with 0xFFFF because indices 0 to 2 have the valid locations of the failing LED safety zones.



FLTSUM_ONGOING	unsigned int	0
FLTSUM_DATAVALID	unsigned int	1
DEV_CNT_LOD_LSD	unsigned int	2
ZONE_CNT_LOD_LSD	unsigned int	3
SZ_CNT_LOD_LSD	unsigned int	3

Figure 3-10. Example Showing Expression chip_status With 3 Faults When Paird With TLC69698-Q1



Variables | Expressions | Registers

Expression	Type	Value
fault_summary	unsigned int[1][64]	[[0x001D,0x0034,0x8031,0xFFFF,0x0000...]] (Hex)
[0]	unsigned int[64]	[0x001D,0x0034,0x8031,0xFFFF,0x0000...] (Hex)
[0]	unsigned int	0x001D (Hex)
[1]	unsigned int	0x0034 (Hex)
[2]	unsigned int	0x8031 (Hex)
[3]	unsigned int	0xFFFF (Hex)

Figure 3-11. Example Showing Expression fault_summary With 3 Faults When Paired With TLC69698-Q1

3.6 Demo

An example of the running demo is presented in this section. Figure 3-12 depicts the TLC69628Q1EVM running in animation mode. Figure 3-13 depicts the TLC69698Q1EVM, Figure 3-14 depicts the TLC69622Q1EVM, and Figure 3-15 depicts the TLC69625Q1EVM.

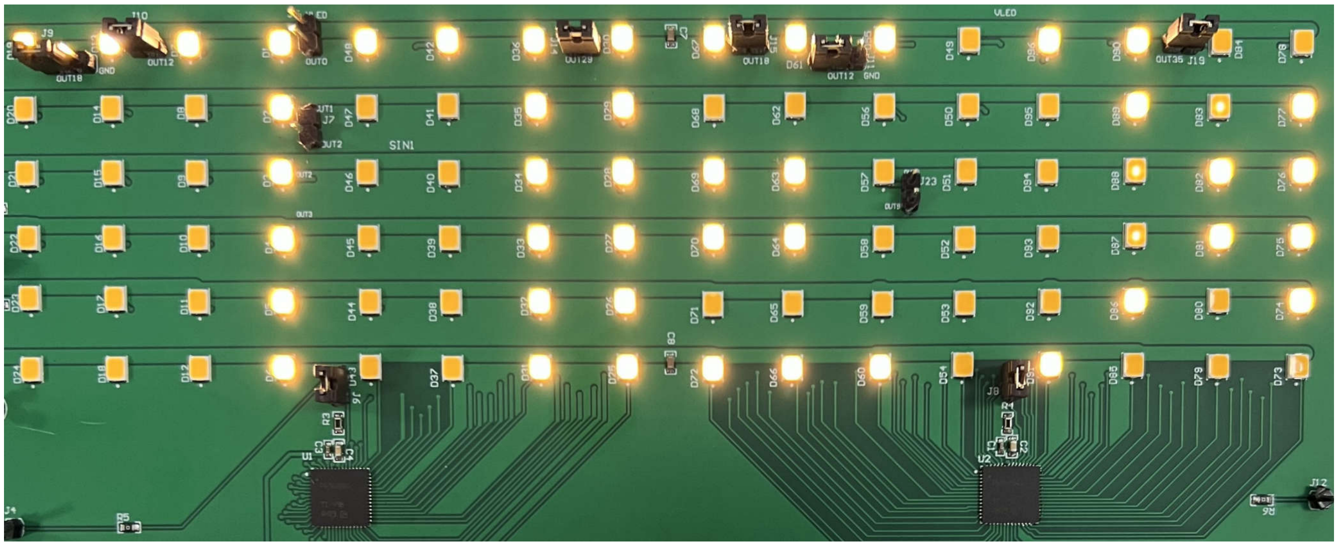


Figure 3-12. Example Demo of TLC69628Q1EVM

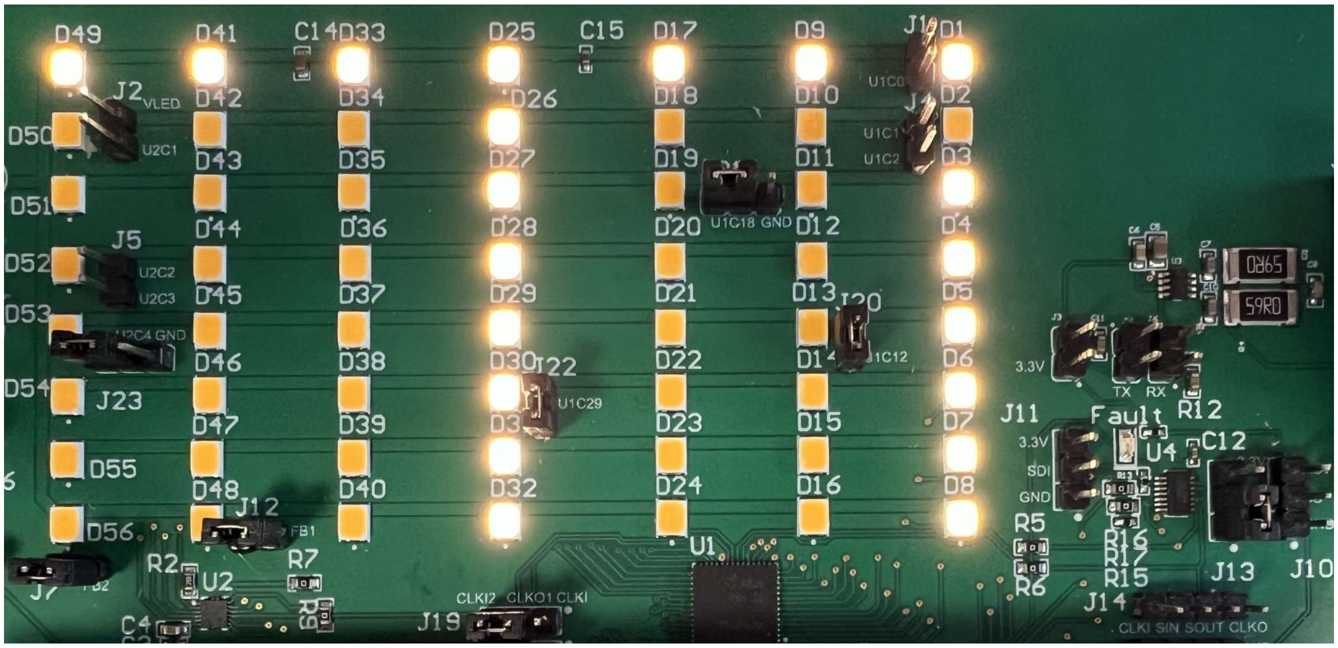


Figure 3-13. Example Demo of TLC69698Q1EVM

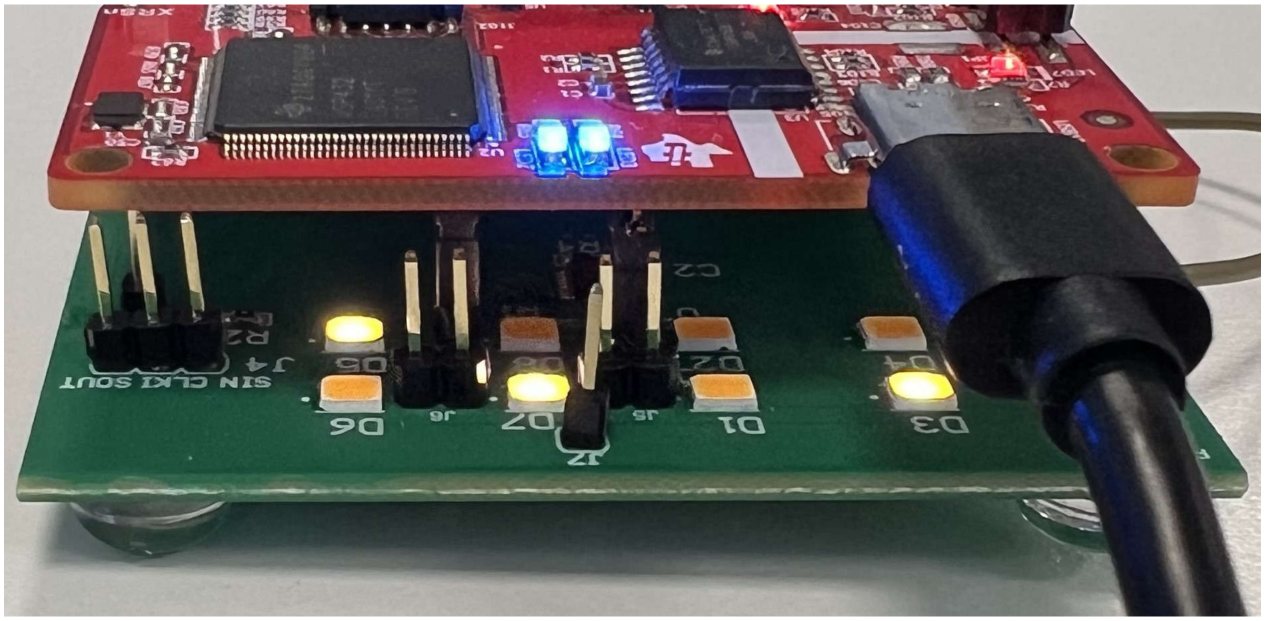


Figure 3-14. Example Demo of TLC69622Q1EVM

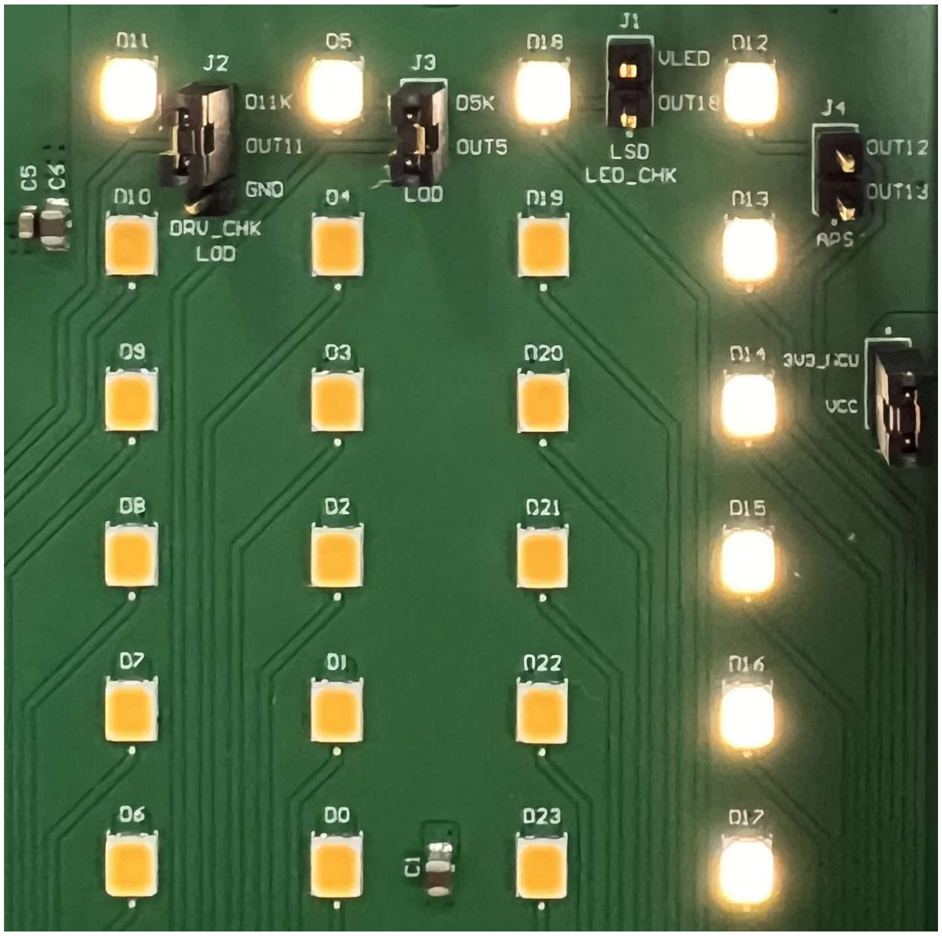


Figure 3-15. Example Demo of TLC69625Q1EVM

4 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

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

- Updated Abstract....................1
- Updated Introduction....................2
- Updated Design Parameters....................5
- Updated Basic System Setup....................7
- Updated Diagnostics....................11
- Updated Demo....................18