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.

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. There are no modifications needed by the user to be able to light up the EVM.

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. The frame is a Texas Instruments logo of 99x6 pixels. This means that not always 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 and LED driver 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.

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.

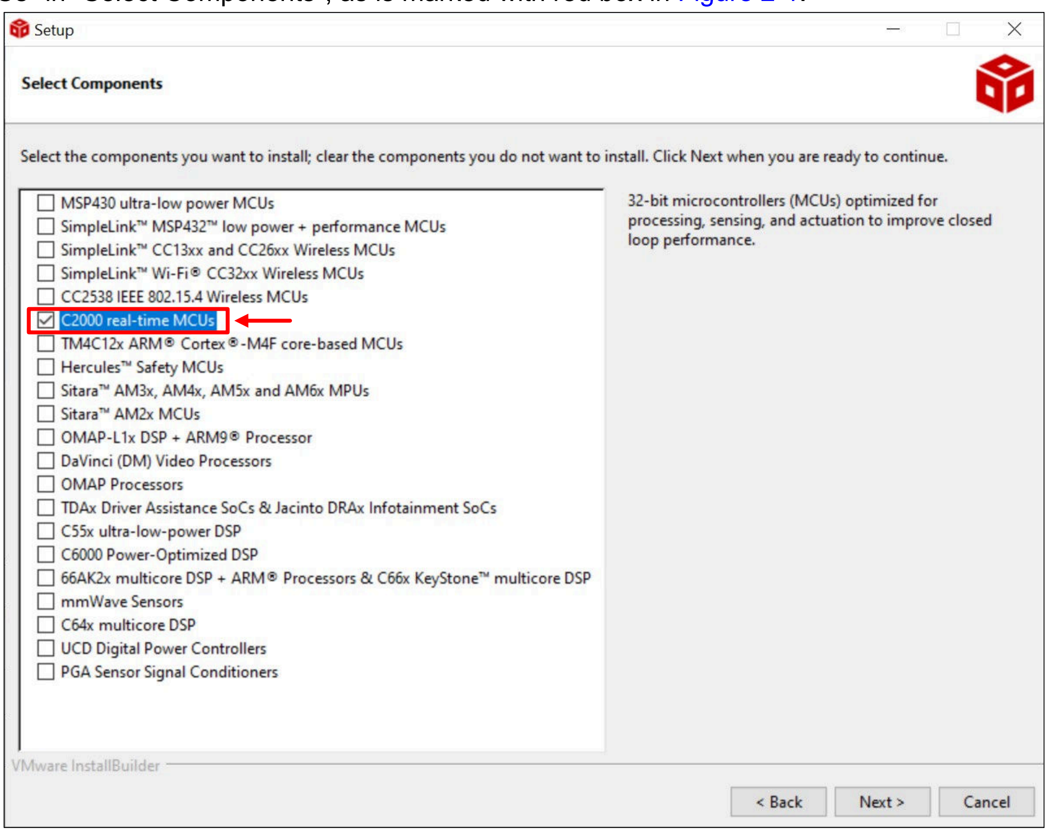


Figure 2-1. Installation Process of Code Composer Studio

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 will show. From the left side bar, select "Code Composer Studio" -> "Products".
    b. Make sure that there is "C2000Ware 5.2.0.00" and "SysConfig" (SysConfig should be 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, you may need to 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.

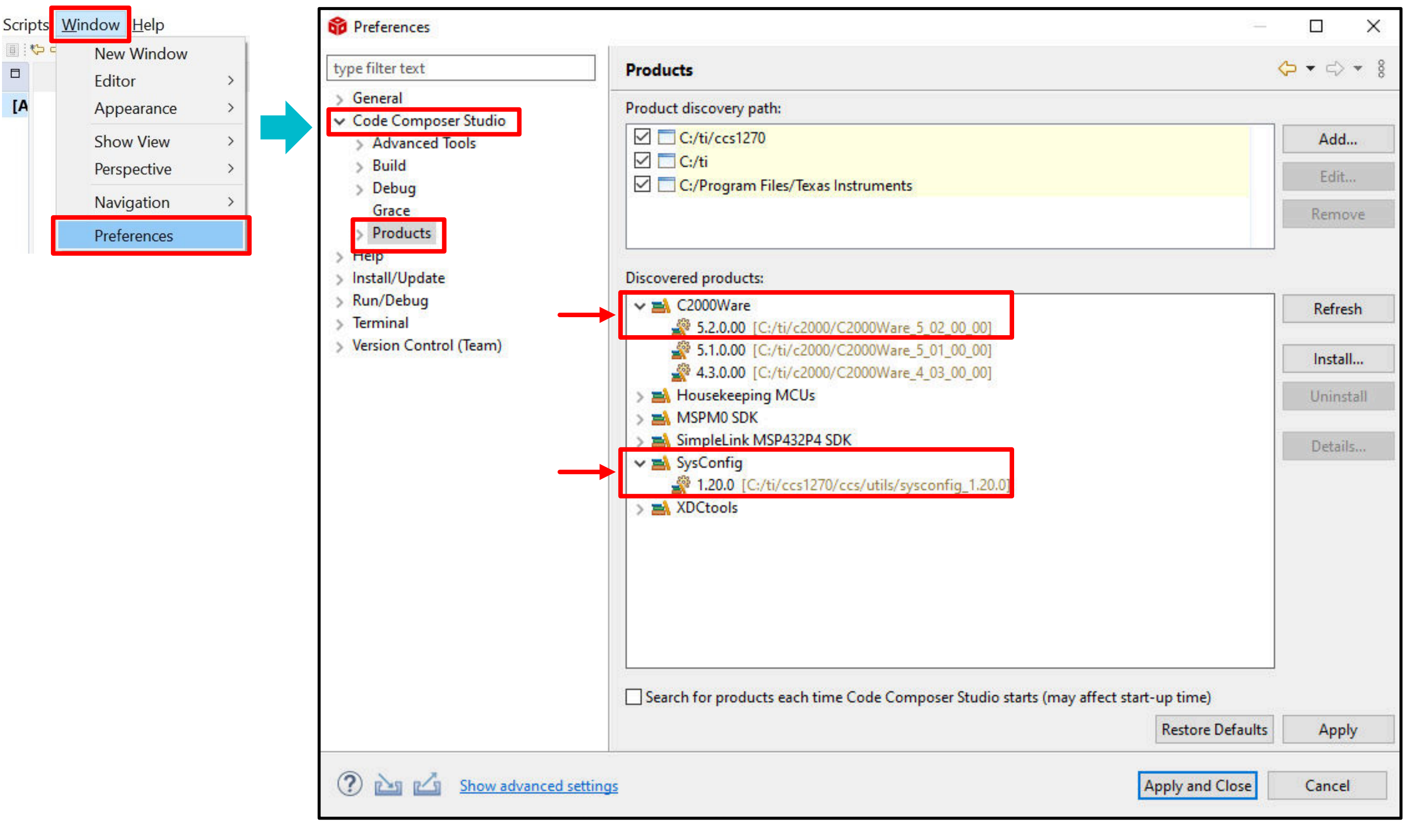


Figure 2-2. Verification of C2000Ware 5.02.0.00 Installation

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, switch S2 on LAUNCHXL-F280039C for both SEL1 and SEL2 should be set 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
Display module size	16 x 6 white LEDs
Frame rate	60Hz
PWM resolution	16 bits
Cascaded devices	2
CLK_I frequency	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`. The MCU setups the clock frequency for the SPI communication to the LED driver.

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.

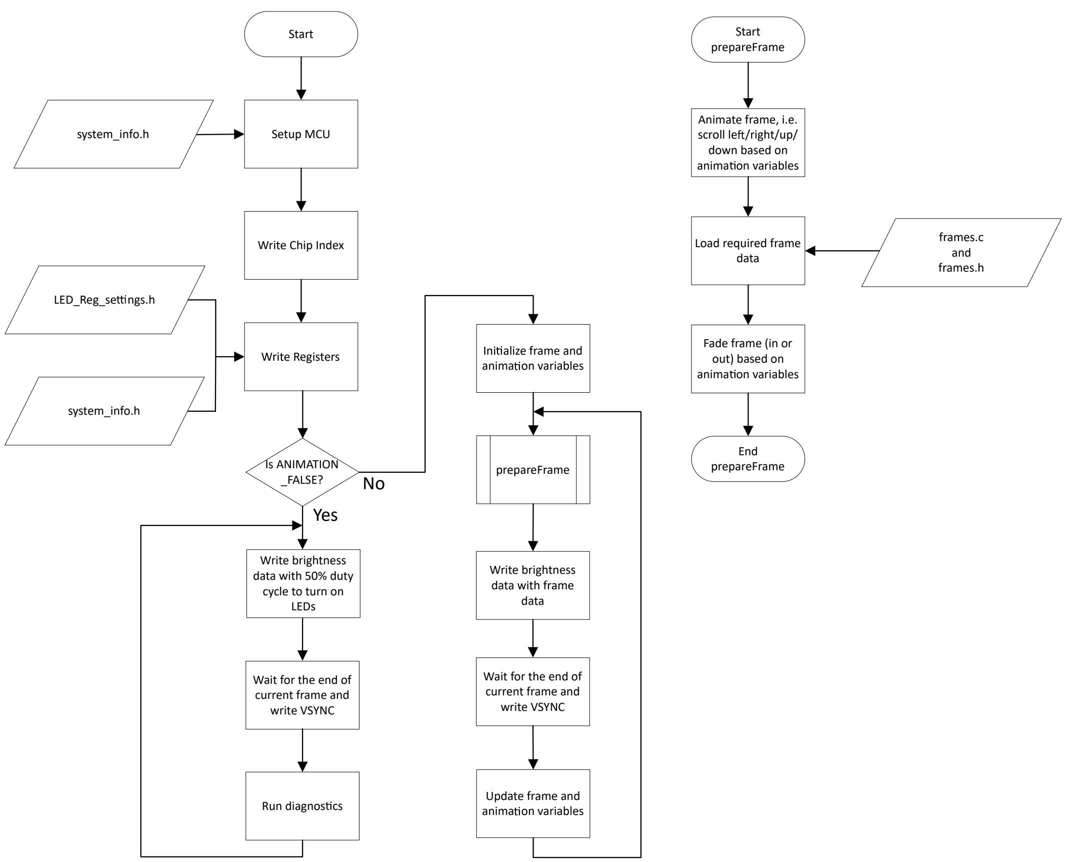


Figure 3-1. Sample Code Flow Diagram

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[7|8]
- TLC6963[7|8]

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 their 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.
	RUN_MODE	Selection between different code run modes. Supported options are *ANIMATION* and *SIMPLE_TEST*.
	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.
	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.

Within the file `system_info.c` the frame rate is specified. The frame period is specified in Hz (frames per second).

```
uint32_t FRAME_RATE = 60;      // 60Hz frames-per-second
```

The minimum supported frame rate is 1Hz.

File `system_info.h` includes several system definitions.

```
// Desired CLK_I frequency
// 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          3000000
```

Macro *SPI_FREQ_IN_HZ* defines at what data rate the SPI is running, such that, the clock frequency of pin CLK_I. This frequency is an integer divider from the system frequency of the MCU. Therefore, the actual SPI frequency may differ from the desired specified frequency defined by *SPI_FREQ_IN_HZ*.

```
#define RUN_MODE                ANIMATION
```

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

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 there are two devices cascaded.

3.4 Advanced System Setup

To describe the advanced system setup, an example is used that is depicted in Figure 3-2.

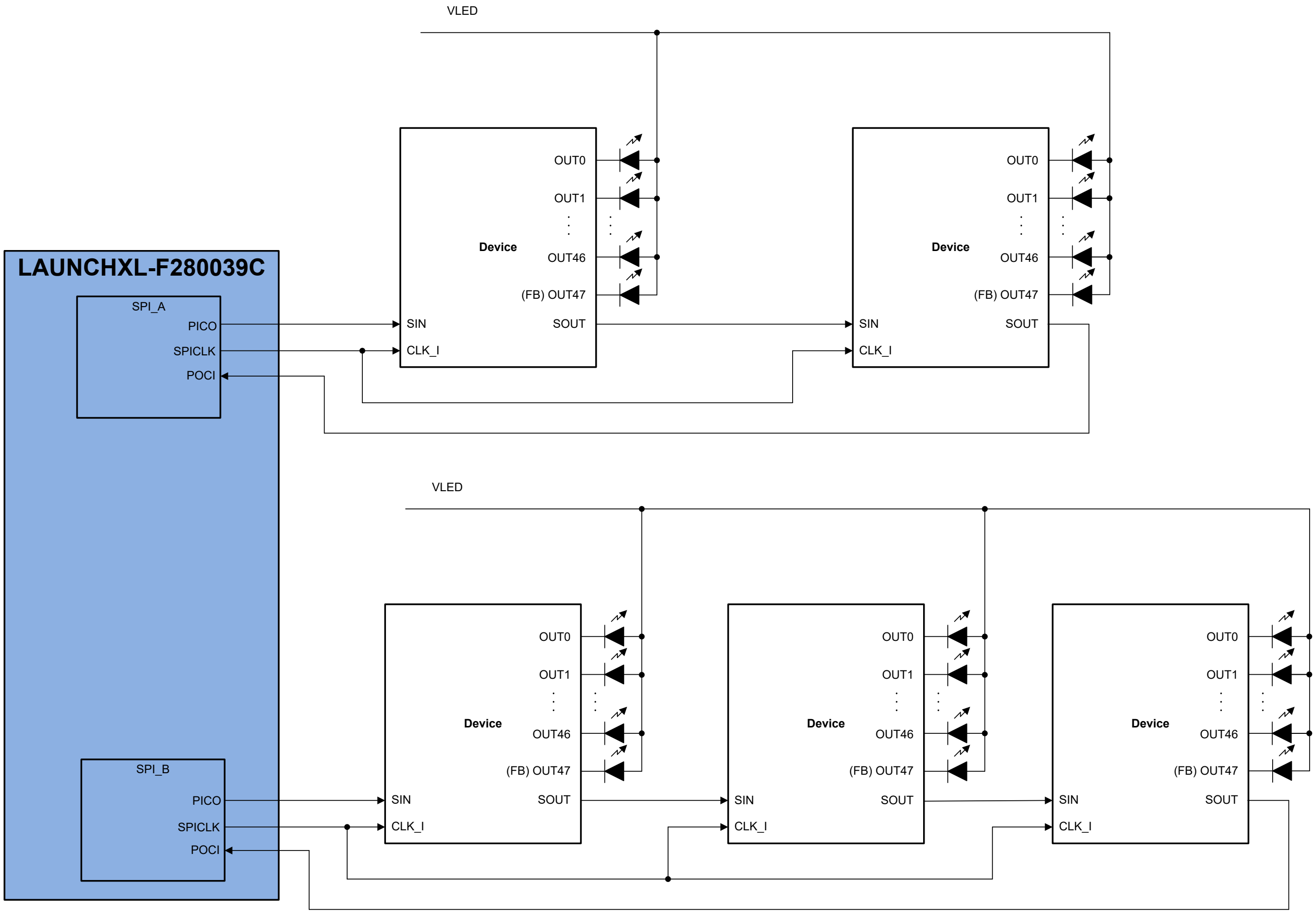


Figure 3-2. Advanced System Example With 2 Daisy Chains

The sample code supports up to two daisy chains. 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* that was described in Section 3.3, there is also macro *CASCADED_UNITS_BUS2* in file `system_info.h`. In the example, chain 1 has two cascaded devices and chain 2 has three cascaded devices.

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

3.5 Diagnostics

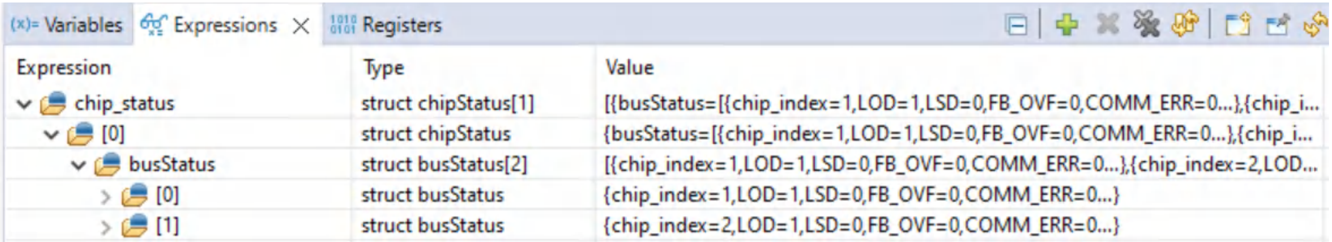
The sample code provides an API to detect which LED driver devices have faults such as Thermal Shutdown (TSD), LED open (LOD), LED short (LSD), Short-to-Ground (STG), Short-to-Power (STP), and Adjacent Pin Short (APS). For the LOD, LSD, STG, STP, and APS, the failing channel is also detected. Within 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`.

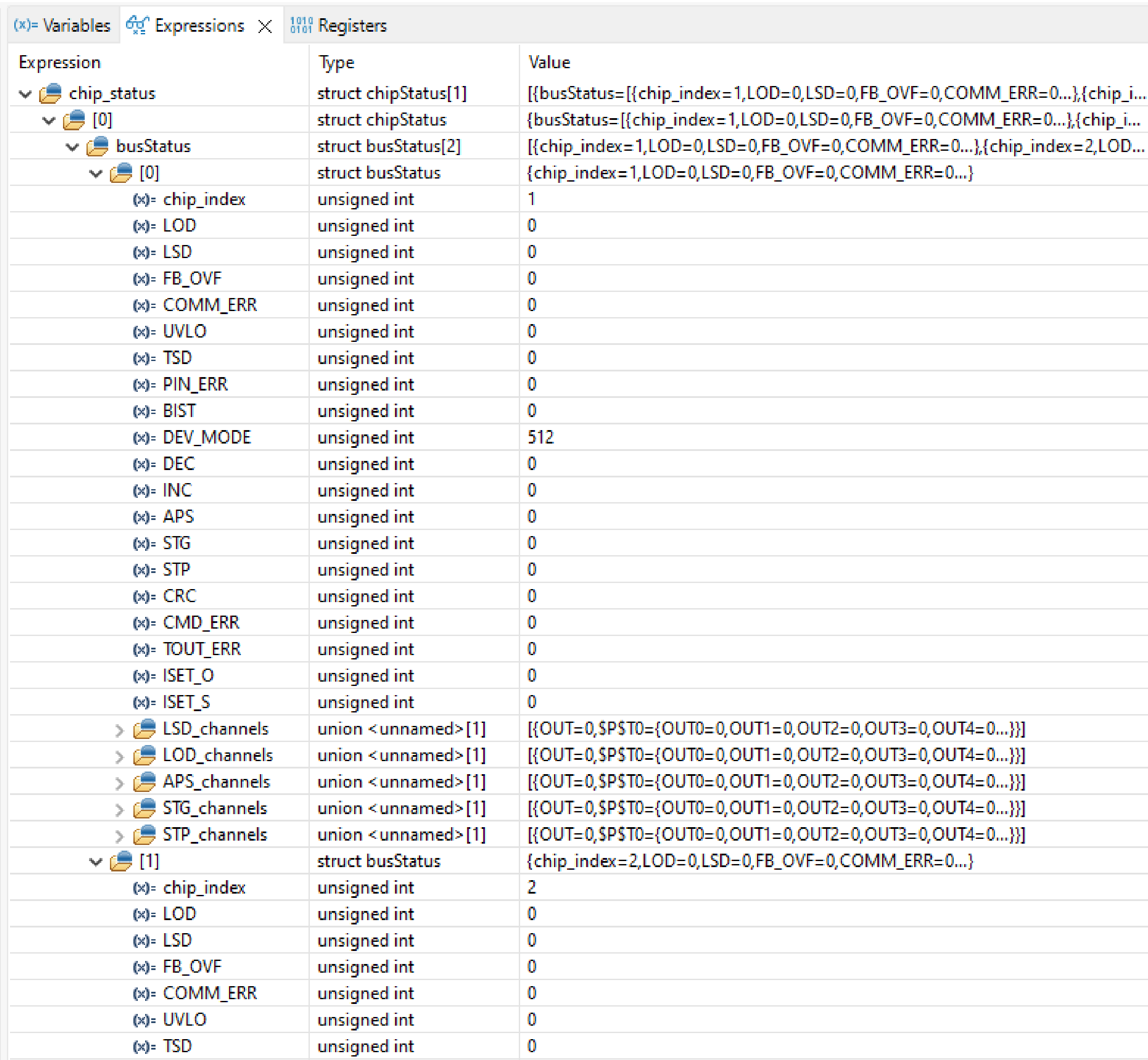
```
struct busStatus {
    uint16_t chip_index;      // Show Chip Index
    uint16_t LOD;             // LED Open Detection
    uint16_t LSD;             // LED Short Detection
    uint16_t FB_OVF;          // Device FB overflow flag
    uint16_t COMM_ERR;        // Device communication error flag including 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, STP, STG, 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 STG;             // Device Short To Ground flag
    uint16_t STP;             // Device Short To Power flag
    uint16_t CRC;             // Device CRC flag
#if defined(_TLC696X8)
    uint16_t COMM_WDOG;       // Device COMM_WDOG flag
#endif
    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
#if defined(_TLC696X8)
    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
#endif
    volatile OUTx LSD_channels[MAX_SCAN_LINES];
    volatile OUTx LOD_channels[MAX_SCAN_LINES];
    volatile OUTx APS_channels[MAX_SCAN_LINES];
    volatile OUTx STG_channels[MAX_SCAN_LINES];
    volatile OUTx STP_channels[MAX_SCAN_LINES];

};

struct chipStatus {
    struct busStatus busStatus[MAX_CASCADED_UNITS];
};

// 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 3-3. The first index of variable *chip_status* is the daisy chain index. On the EVM there only 1 chain is used. 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 EVM there are 2 devices cascaded and, therefore, busStatus runs from 0 to 1.



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...}

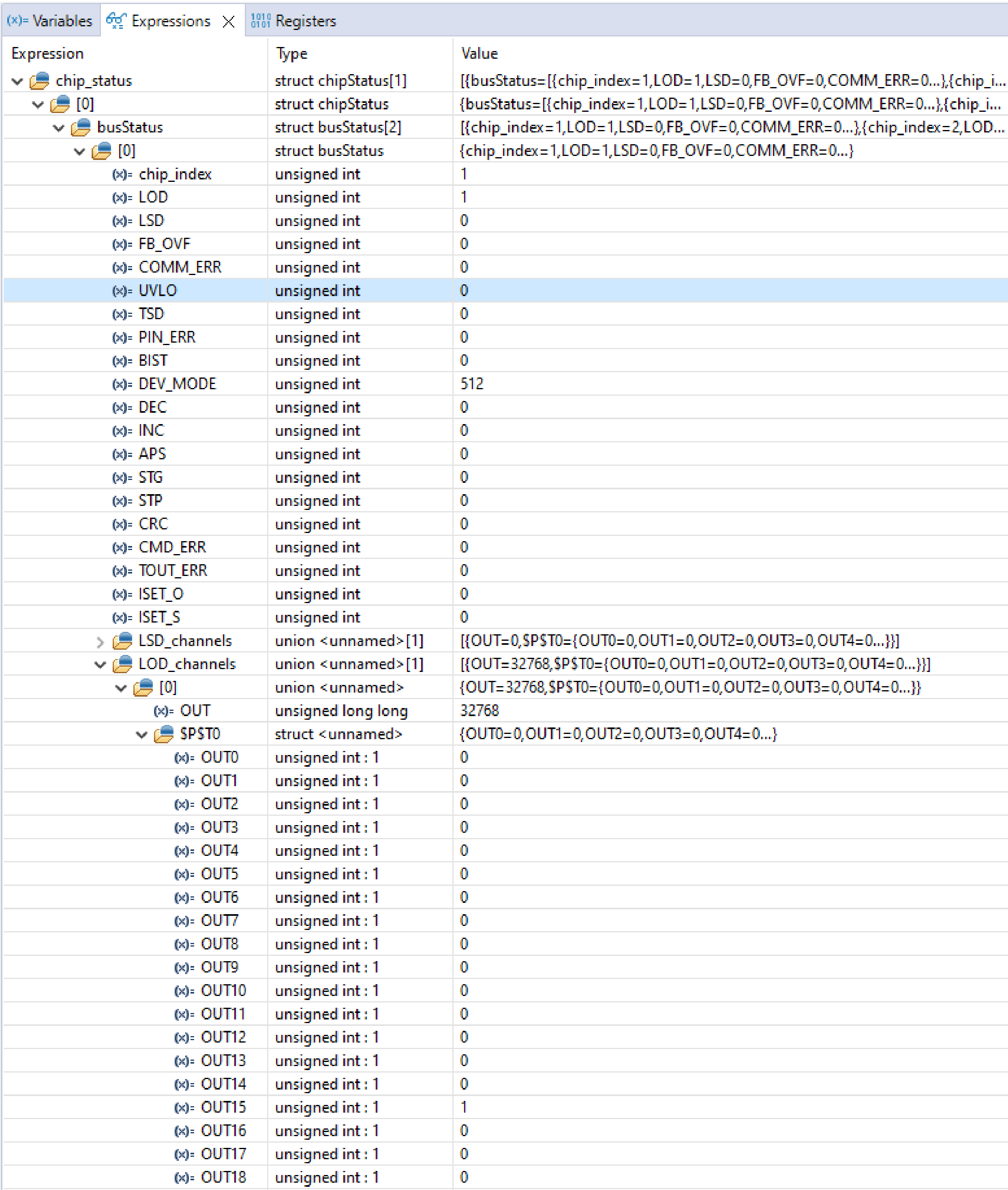
Figure 3-3. Example of Watching Expression chip_status Without Errors



Expression	Type	Value
chip_status	struct chipStatus[1]	[{busStatus=[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_i...
[0]	struct chipStatus	{busStatus=[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_i...
busStatus	struct busStatus[2]	[{chip_index=1,LOD=0,LSD=0,FB_OVF=0,COMM_ERR=0...},{chip_index=2,LOD...
[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	512
DEC	unsigned int	0
INC	unsigned int	0
APS	unsigned int	0
STG	unsigned int	0
STP	unsigned int	0
CRC	unsigned int	0
CMD_ERR	unsigned int	0
TOUT_ERR	unsigned int	0
ISET_O	unsigned int	0
ISET_S	unsigned int	0
LSD_channels	union <unnamed>[1]	[{OUT=0,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}]
LOD_channels	union <unnamed>[1]	[{OUT=0,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}]
APS_channels	union <unnamed>[1]	[{OUT=0,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}]
STG_channels	union <unnamed>[1]	[{OUT=0,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}]
STP_channels	union <unnamed>[1]	[{OUT=0,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}]
[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
COMM_ERR	unsigned int	0
UVLO	unsigned int	0
TSD	unsigned int	0

Figure 3-4. Expanded Example of Watching Expression chip_status Without Errors

Figure 3-4 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, STG, STP, and APS faults the actual output channel that has the fault can be found. An example of LOD fault is depicted in Figure 3-5. In this example, chip index 1 (which is index 0 of busStatus) has an LOD fault. When array LOD_channels is expanded, it shows that the fault occurs on pin OUT15. 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...},{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...}
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	512
DEC	unsigned int	0
INC	unsigned int	0
APS	unsigned int	0
STG	unsigned int	0
STP	unsigned int	0
CRC	unsigned int	0
CMD_ERR	unsigned int	0
TOUT_ERR	unsigned int	0
ISET_O	unsigned int	0
ISET_S	unsigned int	0
LSD_channels	union <unnamed>[1]	[{OUT=0,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}]
LOD_channels	union <unnamed>[1]	[{OUT=32768,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}]
[0]	union <unnamed>	{OUT=32768,PT0={OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}}
OUT	unsigned long long	32768
PT0	struct <unnamed>	{OUT0=0,OUT1=0,OUT2=0,OUT3=0,OUT4=0...}
OUT0	unsigned int : 1	0
OUT1	unsigned int : 1	0
OUT2	unsigned int : 1	0
OUT3	unsigned int : 1	0
OUT4	unsigned int : 1	0
OUT5	unsigned int : 1	0
OUT6	unsigned int : 1	0
OUT7	unsigned int : 1	0
OUT8	unsigned int : 1	0
OUT9	unsigned int : 1	0
OUT10	unsigned int : 1	0
OUT11	unsigned int : 1	0
OUT12	unsigned int : 1	0
OUT13	unsigned int : 1	0
OUT14	unsigned int : 1	0
OUT15	unsigned int : 1	1
OUT16	unsigned int : 1	0
OUT17	unsigned int : 1	0
OUT18	unsigned int : 1	0

Figure 3-5. Example of chip_status With LED Open (LOD) Fault

3.6 Demo

An example of the running demo is presented in this section. Figure 3-6 depicts the TLC69628Q1EVM running in animation mode.

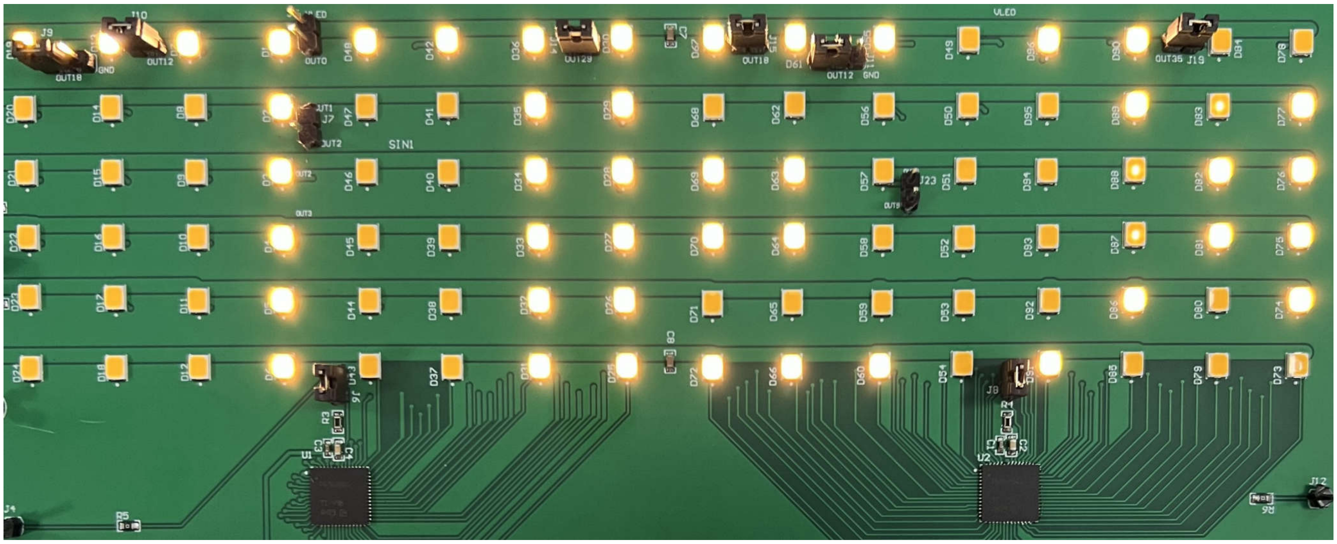


Figure 3-6. Example Demo of TLC69628Q1EVM

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265