User's Guide

LP589x-Q1 Sample Code

ABSTRACT

This document describes the preparation and usage of the sample code for the LP589x-Q1 device family when paired with a LAUCHXL-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

1 Introduction

The sample code showcases the ability to light up the LEDs on the LP5891Q1EVM.

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 two frames are used to scroll left, right, up, and down and to fade in and fade out according to a predefined sequence. The first frame is a Texas Instruments logo of 32x32 RGB pixels and the second frame a rainbow pattern of 48x32 RGB 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 3.5. It is outside the scope of this document to explain how the frames in the sample code are generated.

In the simple test mode, the user can use some predefined APIs to light up the LED board, perform diagnostics, or build custom Continuous Clock Serial Interface (CCSI) commands. The sample code comes with turning on all RGB LEDs which results in a white display. In addition, every frame, diagnostics is executed. For more information about diagnostics see Section 3.4.

The predefined APIs automatically adjust to the specified system. More detail about the system specification can be found in Section 3.3.

2 Software Setup

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

1. Download and install Code Composer Studio™.
   a. Download Code Composer Studio integrated development environment (IDE) (Version ≥ 11.1.0).
   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.

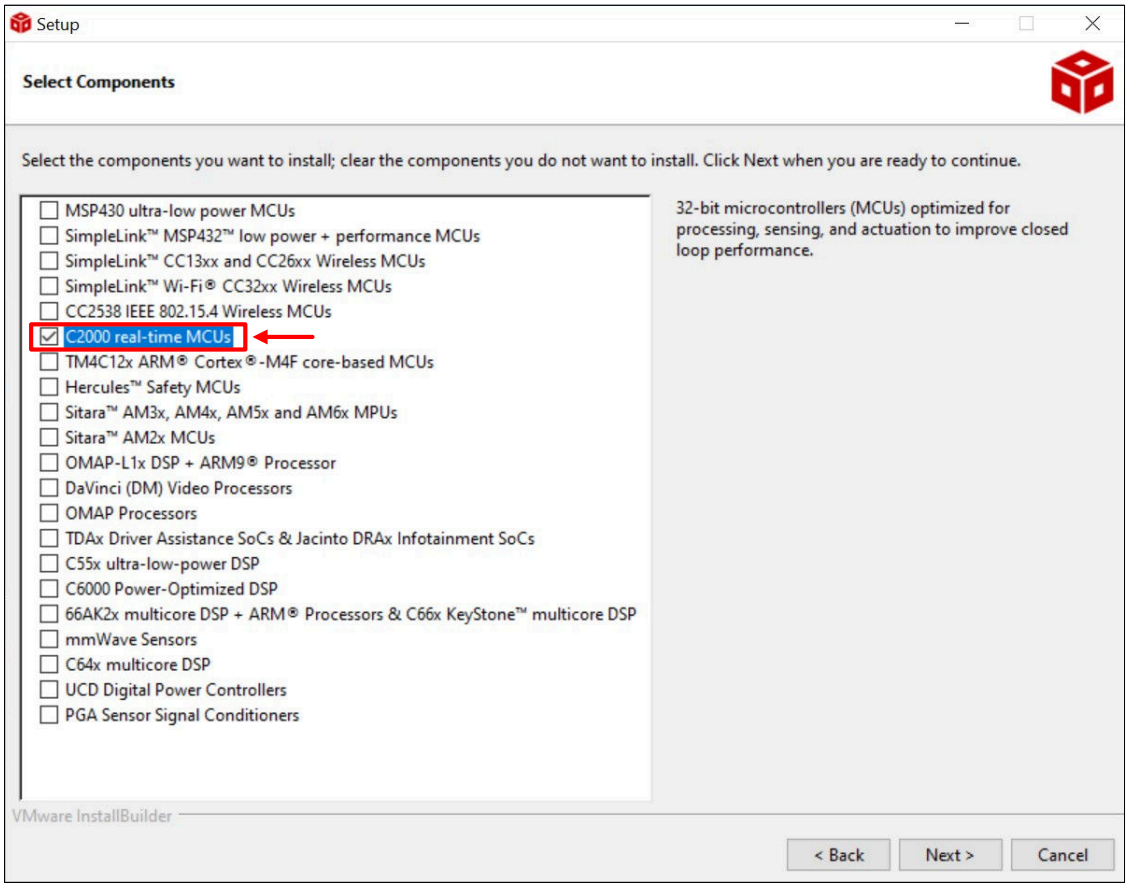

Figure 2-1. Installation Process of Code Composer Studio

2. Download and install C2000Ware (Version ≥ 4.01.00.00).
   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 4.1.0.00" (or later version) 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 4.1.0.00" (or later version) 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.

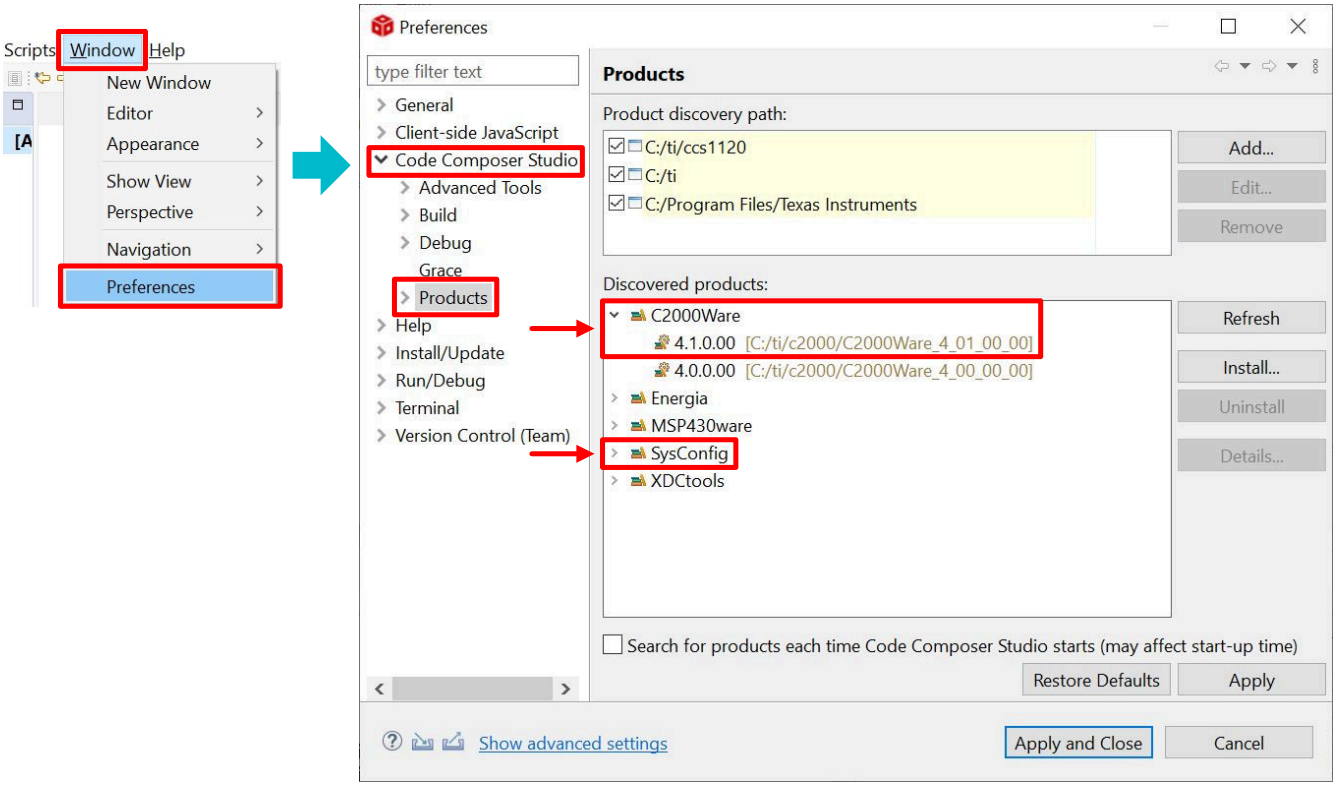

Figure 2-2. Verification of C2000Ware 4.1.0.00 (or later version) Installation

3. Download and import sample code.
   a. The link for each EVM is different. However, the sample code in each link is the same except for the file `led_driver.h` which is setup for the matching EVM.
      i. LP5891Q1EVM: LP5891Q1EVM-SW-F280039C
   b. Importing the Code Composer Studio (CCS) project according to the process provided in the link: Importing a CCS Project.
4. Load the program according to the process provided in the link: Building and Running Your Project.
5. (optional) Download the register map generation tool. This is a handy tool if you want to further configure the registers.
   a. LP5891-Q1: LP5891 Registers Map Generation Tool

3 Sample Code Structure

3.1 Design Parameters

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

Table 3-1. Design Parameters EVMs

Design Parameter	LP589x
Display module size	16 × 16 RGB LEDs
Frame rate	60 Hz
Refresh rate	7680 Hz
PWM resolution	16 bits
Cascaded devices	1
SCLK frequency	7.5 MHz
GCLK frequency	75 MHz

3.2 Flow Diagram

Figure 3-1 depicts the high level flow in the sample code.

During the Setup MCU, the frequency that is used for the Continuous Clock Serial Interface (CCSI) is defined in file `system_info.h`.

The `FC_settings.h` file can automatically be generated by the Registers Map Generation Tool mentioned in Section 2. Not all register settings are coming from this file. The number of scan lines (field SCAN_NUM in register FC0) and the number of cascaded devices (field CHIP_NUM in register FC0) are coming from the file `system_info.h`. This file is described in more detail in Section 3.3.

The flow diagram also shows the files `frames.c` and `frames.h` which contain the 2 frames used during the animation mode. It is outside the scope of this document to explain how these frames are generated.

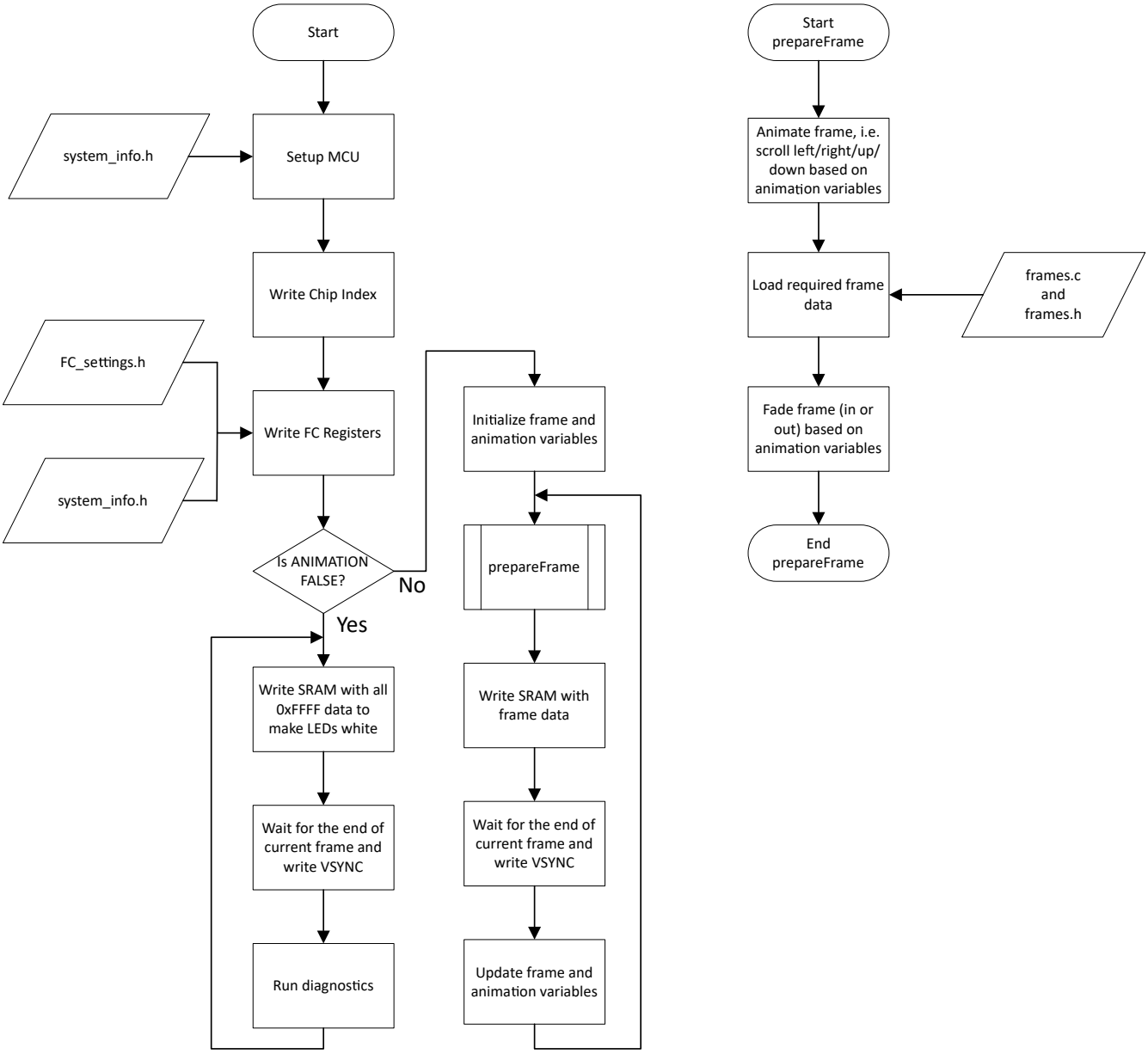

Figure 3-1. Sample Code Flow Diagram

3.3 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 used LED driver IC is selected.

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

The code supports:

- LP5891
- LP5890
- TLC6984
- TLC6983

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	*SPICLK_FREQ_IN_HZ*	SCLK frequency
	ANIMATION	Selection between animation and simple test modes
	TOTAL_SCAN_LINES	Number of scan lines
	CCSI_BUS_NUM	Number of CCSI busses
	CASCADED_UNITS_CCSI1	Device count in CCSI bus 1
	CASCADED_UNITS_CCSI2	Device count in CCSI bus 2
	MONOCHROMATIC	Selection between RGB and single color display
system_info.c	*FRAME_PERIOD*	Interval of VSYNC commands

Within the file system_info.c the frame period is specified which determines the frame rate. The frame period is specified in microseconds.

```
const uint16_t FRAME_PERIOD = 16667;      // 16.67ms = 60 Hz frames-per-second
```

The maximum supported frame period is 65535 microseconds, i.e. the lowest frame rate is 15.3 Hz.

File system_info.h includes several system definitions.

```
// Desired SCLK frequency (in case of TLC698x this SCLK frequency is half of this)
// Note: Exact frequency may not be possible
#define SPICLK_FREQ_IN_HZ        7500000
```

Macro *SPICLK_FREQ_IN_HZ* defines at what data rate the Continuous Clock Serial Interface (CCSI) is running, i.e. the clock frequency of pin SCLK. This frequency is an integer divider from the system frequency of the MCU. Therefore, the actual CCSI frequency may differ from the desired specified frequency defined by *SPICLK_FREQ_IN_HZ*.

```
#define ANIMATION              TRUE
#define MONOCHROMATIC          FALSE
```

Macro *ANIMATION* will determine if the animation or simple test mode is executed.

The EVMs all use RGB LEDs. Therefore, macro *MONOCHROMATIC* is defined as FALSE. The sample code does support systems using single color LEDs, e.g. only red LEDs. In those cases, the macro *MONOCHROMATIC* should be defined as TRUE. This automatically changes the frame data structure, the animation algorithms, and APIs.

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

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

Macro *TOTAL_SCAN_LINES* defines the number of scan lines used in the system and will directly impact the field SCAN_NUM in register FC0.

For the LP5891Q1EVM there are 16 scan lines.

Macro *CASCADED_UNITS_CCSI1* defines the number of cascaded devices in the system and directly impacts the field CHIP_NUM is register FC0.

For the LP5891Q1EVM there is only 1 device cascaded.

When the user cascades more EVMs with the available connectors, this macro will have to be updated.

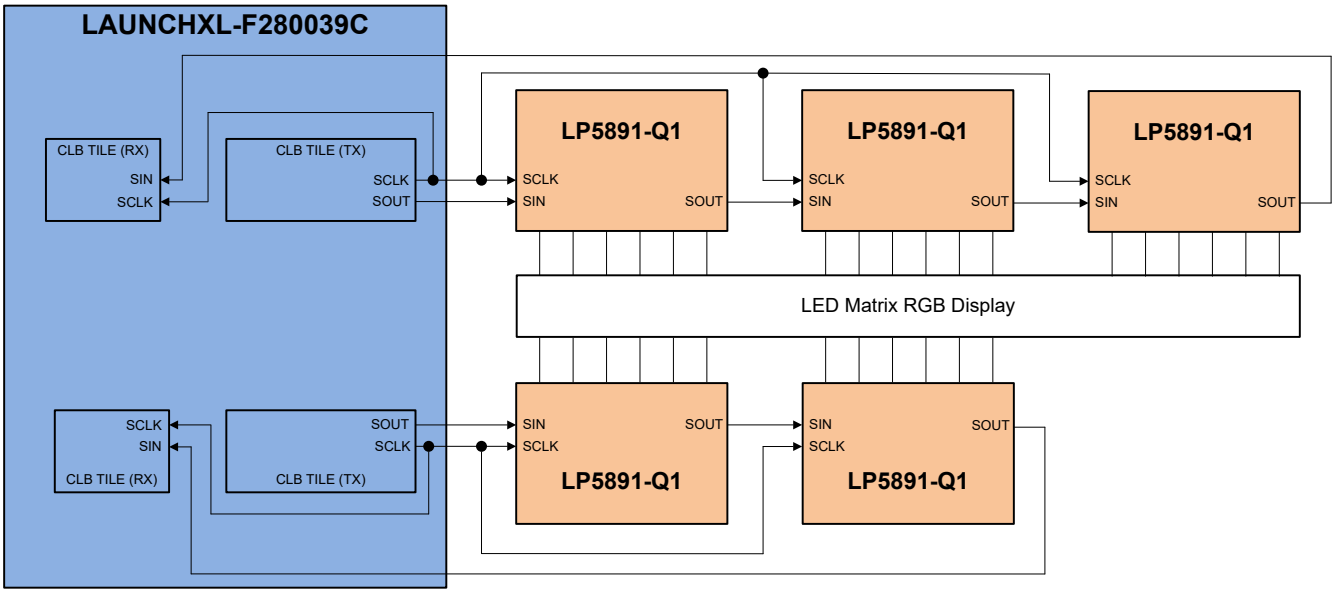

Figure 3-2. Example System Using 2 CCSI Chains

The sample code supports up to 2 CCSI daisy chains. To illustrate this an example is depicted in Figure 3-2. The number of actual used chains is defined by macro *CCSI_BUS_NUM* in file `system_info.h`.

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

Each chain can have a different number of cascaded devices. Therefore, besides macro *CASCADED_UNITS_CCSI1*, there is also macro *CASCADED_UNITS_CCSI2* in file `system_info.h`. In the example, CCSI chain 1 has 3 cascaded devices and CCSI chain 2 has 2 cascaded devices.

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

3.4 Diagnostics

The sample code provides an API to detect which devices have LED open (LOD) and which devices have LED short (LSD) faults. Within the `TLC698x_LP589x_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 `system_info.h`.

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

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

Note that in the sample code the diagnostics is only executed during simple test mode and not during animation mode.

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 CCSI chain index. On the EVM there only 1 chain is used. The second index is the chip index.

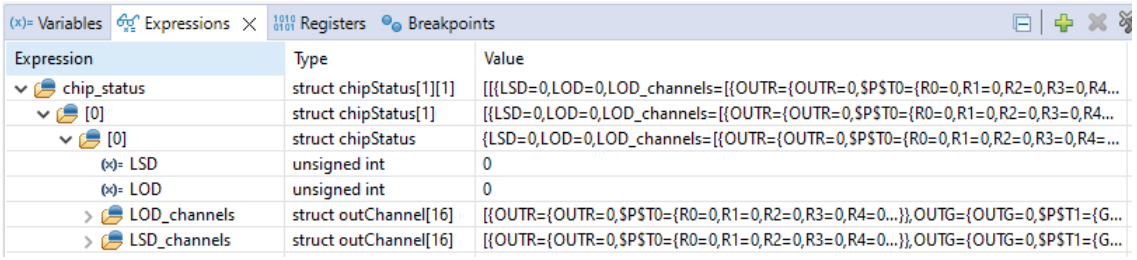

Figure 3-3. Example of Watching Expression chip_status Without Errors

Figure 3-4 depicts an expanded view of the *chip_status* variable. For each device in the chain, the LSD and LOD are shown. In addition, for the LSD and LOD faults the actual line and output channel which has the fault can be found. The LP5891Q1EVM uses 16 scan lines. Therefore, the indices for array LOD_channels run from 0 to 15.

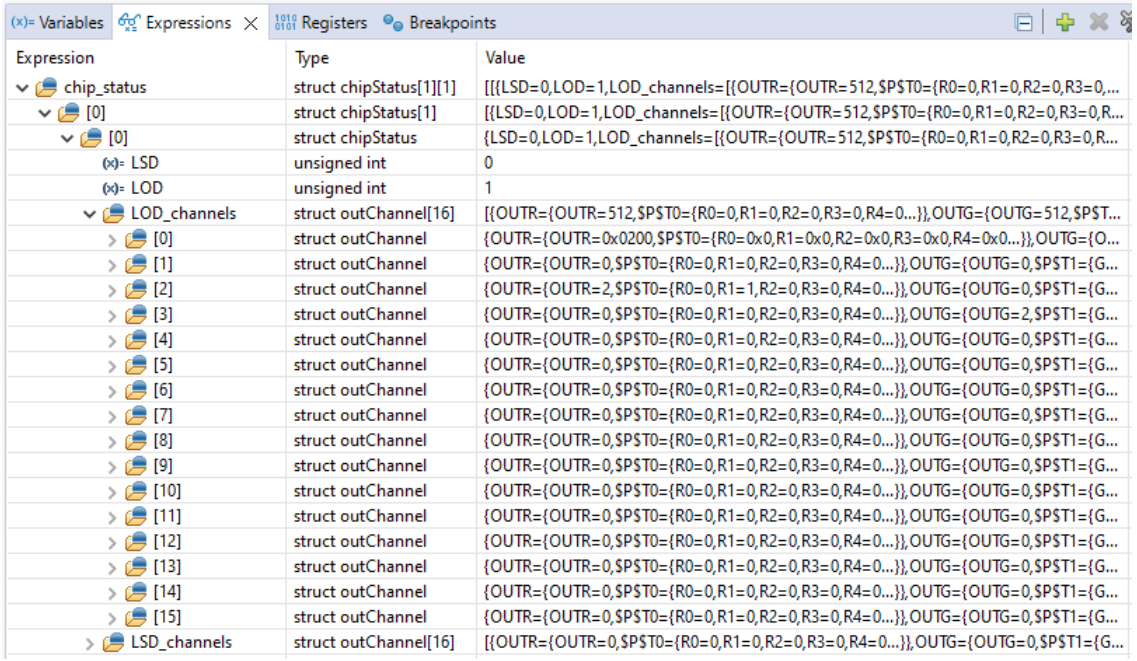

Figure 3-4. Example Showing Expanded Scan Lines of Expression chip_status With LOD

An example of LOD fault is depicted in Figure 3-5. In this example chip index 0 has an LOD fault. When array LOD_channels is expanded, it shows that the fault occurs on line with index 2. When that index is expanded, it shows that the fault happens on pin R1.

Expression	Type	Value
chip_status	struct chipStatus[1][1]	[[{LSD=0,LOD=1,LOD_channels=[{OUTR={OUTR=512,PT0={R0=0,R1=0,R2=0,R3=0,...
[0]	struct chipStatus[1]	[{LSD=0,LOD=1,LOD_channels=[{OUTR={OUTR=512,PT0={R0=0,R1=0,R2=0,R3=0,R...
[0]	struct chipStatus	{LSD=0,LOD=1,LOD_channels=[{OUTR={OUTR=512,PT0={R0=0,R1=0,R2=0,R3=0,R...
LSD	unsigned int	0
LOD	unsigned int	1
LOD_channels	struct outChannel[16]	[{OUTR={OUTR=512,PT0={R0=0,R1=0,R2=0,R3=0,R4=0...}},OUTG={OUTG=512,PT...
[0]	struct outChannel	{OUTR={OUTR=0x0200,PT0={R0=0x0,R1=0x0,R2=0x0,R3=0x0,R4=0x0...}},OUTG={O...
[1]	struct outChannel	{OUTR={OUTR=0,PT0={R0=0,R1=0,R2=0,R3=0,R4=0...}},OUTG={OUTG=0,PT1={G...
[2]	struct outChannel	{OUTR={OUTR=2,PT0={R0=0,R1=1,R2=0,R3=0,R4=0...}},OUTG={OUTG=0,PT1={G...
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=2,PT1={G...

Figure 3-5. Example Showing Expanded Channels of Expression chip_status With LOD

3.5 Demo

In this section, examples of the LED demo are presented. Figure 3-6 depicts the LP5891Q1EVM running a demo.

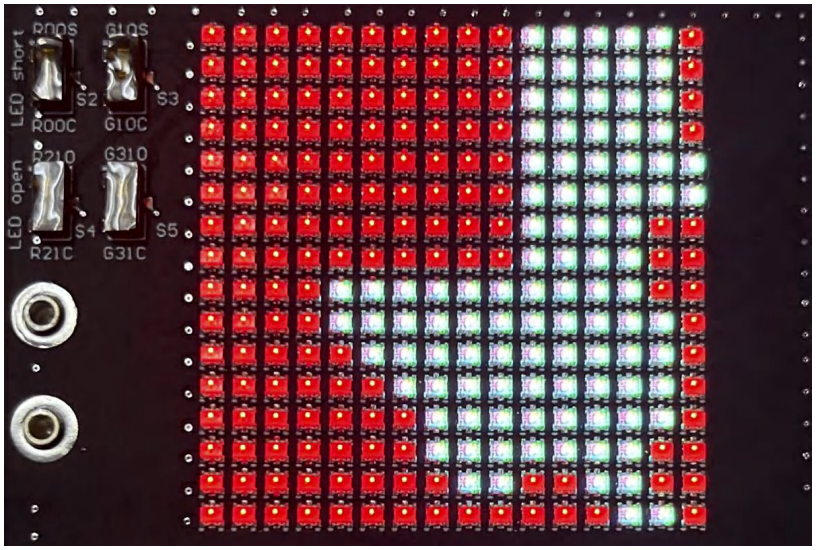

Figure 3-6. LP5891Q1EVM Demo Example

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