



摘要

本文档可作为准备和使用与 MSP-EXP430F5529LP 配对的 TPS929xxx-Q1 器件系列示例代码的指南。可从网站下载的代码将能够点亮每个匹配 EVM 的 LED。

内容

1 引言.....	2
2 软件设置.....	2
3 硬件设置.....	3
4 示例代码结构.....	4
4.1 流程图.....	4
4.2 系统设置.....	5
4.3 诊断.....	6
4.4 EEPROM 编程.....	9

插图清单

图 2-1. Code Composer Studio 安装过程.....	2
图 3-1. MSP-EXP430F5529LP 上的连接.....	4
图 4-1. 示例代码流程图.....	5
图 4-2. 观察 TPS929120-Q1 的表达式 chip_status (无错误) 的示例.....	7
图 4-3. 观察 TPS929120-Q1 的表达式 chip_status (具有短路故障) 的示例.....	8
图 4-4. 观察 TPS929120-Q1 的表达式 chip_status (具有低电源) 的示例.....	8

表格清单

表 3-1. 硬件连接.....	3
表 4-1. 每个文件的宏和变量名称的摘要.....	6
表 4-2. 在 EEPROM 编程期间使用 REF 引脚时要设置的 EVM 跳线.....	9

商标

LaunchPad™ and Code Composer Studio™ are trademarks of Texas Instruments.

所有商标均为其各自所有者的财产。

1 引言

示例代码展示了点亮 TPS929120EVM、TPS929160EVM 和 TPS929240EVM 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 EEPROM 编程。默认选择动画模式。节 4.2 介绍了如何在这两种模式之间切换。在动画模式下，根据预定义的顺序使用 6 种不同的动画。通过点击 MSP-EXP430F5529LP 上的按钮 S2，可在不同动画之间切换。在每次播放动画后，都会执行诊断以确定是否发生了任何故障。更多有关诊断结果的信息，请参阅节 4.3。

示例代码附带许多预定义的 API，这些 API 可用于更改 LED 驱动器的配置、执行诊断或构建自定义 FlexWire 命令。预定义的 API 会根据指定的系统自动调整。如需了解有关系统演示的更多详情，请参阅节 4.2。

2 软件设置

要为 MSP430F5529 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 ≥ 11.1.0)。
 - 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“MSP430 ultra-low power MCUs”，如图 2-1 中的红色框所示。

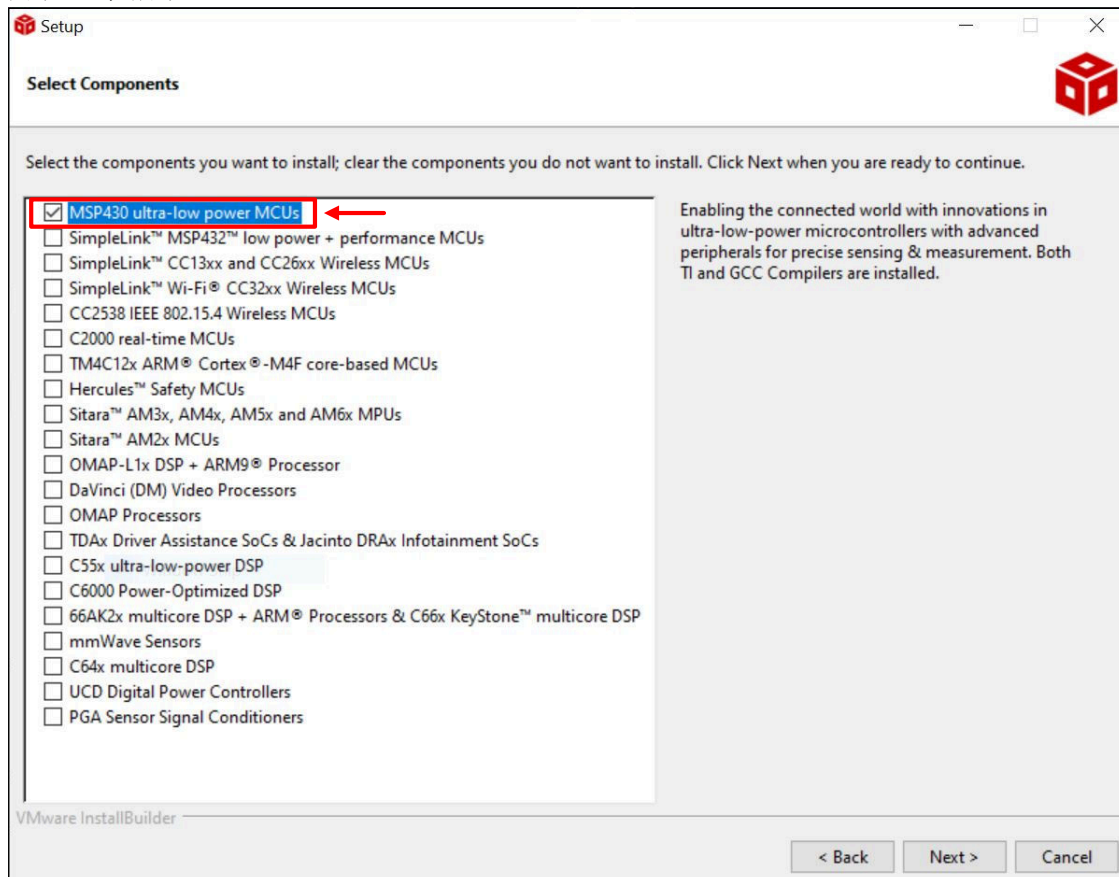


图 2-1. Code Composer Studio 安装过程

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 文件之外，每个链接中的示例代码都是相同的。
 - i. TPS929120EVM : [TPS92912XQ1-SW-F5529](#)
 - ii. TPS929160EVM : [TPS929160Q1-F5529-SW](#)
 - iii. TPS929240EVM : [TPS929240Q1-F5529-SW](#)
3. 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 EEPROM 配置工具。如果您想使用非默认数据对 EEPROM 进行编程，这是一个很方便的工具。TPS929160/TPS929240 的工具还包括一个计算工具，用于计算选项卡 IF_CRC 中的 FlexWire 接口 CRC 值。对于每个受支持的器件，都有一个单独的链接 :
 - a. TPS929120/TPS929121 : [TPS92912x-Q1 EEPROM 配置工具](#)
 - b. TPS929160/TPS929240 : [TPS929240-Q1 TPS929160-Q1 EEPROM 配置工具](#)

3 硬件设置

本节介绍了硬件设置与每个器件专用 EVM 用户指南中的描述之间的差异。检查以下指南中的硬件设置 :

- [TPS929120EVM 用户指南](#)
- [TPS929160EVM 用户指南](#)
- [TPS929240EVM 用户指南](#)

示例代码将 USB2ANY 替换为 MSP-EXP430F5529LP。将 MSP-EXP430F5529LP 连接到 EVM 有两种方法 :

- UART
- TPS929120CANEVM

对于这两种方法，表 3-1 中列出了连接。此外，图 3-1 描绘了 MSP-EXP430F5529LP 上的位置。除连接外，节 4.1 中详述的示例代码中也使用了开关 S1 (青色) 和 S2 (绿色)。在 EVM 上，必须针对每种模式正确设置跳线。器件专用 EVM 用户指南对此进行了介绍。

表 3-1. 硬件连接

接口	板	UART-RX	UART-TX	+3.3V	GND	+5V
	颜色	蓝色	黄色	紫	黑色	橙色
	MSP-EXP430F5529LP	P3.4 (J1-3)	P3.3 (J1-4)	3V3 (J1-1)	GND (J3-22)	+5V (J3-21)
CAN	TPS929120CANEVM	J3-13	J3-14	J3-15	J3-16	J3-3
UART	TPS929120EVM	J3-3	J3-4	J3-5	J3-6	不需要
	TPS929160EVM	J29-3	J29-4	J29-5	J29-6	
	TPS929240EVM	J4-3	J4-4	J4-5	J4-6	

这些连接应通过手接方式连接。

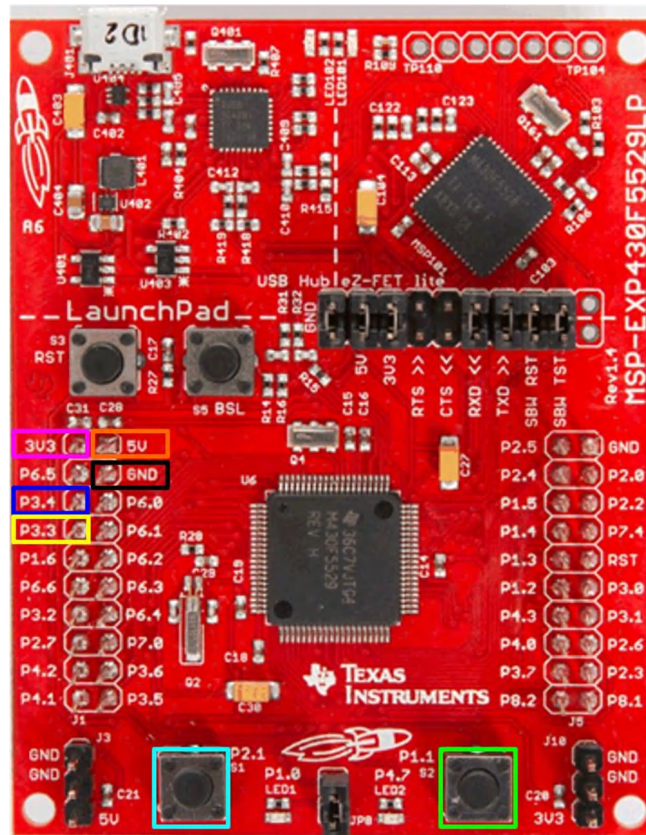


图 3-1. MSP-EXP430F5529LP 上的连接

4 示例代码结构

4.1 流程图

图 4-1 描述了示例代码中的高级流程。在整个流程中，使用了 FlexWire 总线上的器件数量及其地址。system_info.h 和 system_info.c 文件中指定了这些信息，节 4.2 中对此进行了更详细的说明。

设置 MCU 会配置 UART 接口并将其设置为 750000 波特。解锁 LED 驱动器后，将对其进行检查，以确定是否选择了动画模式或 EEPROM 编程模式。节 4.2 介绍了如何在这两种模式之间切换。在动画模式期间，系统将执行 LED 图形，并在完成后检查诊断结果。如需详细了解诊断结果，请参阅节 4.3。诊断完成后，将对其进行检查，以确定是否按下了 MSP-EXP430F5529LP 上的按钮 S2。如果未按下该按钮，将再次执行同一 LED 图形。如果按下了该按钮，将执行下一个 LED 图形，直到所有 6 个图形都已执行，循环将再次从第一个图形重新开始。

在 EEPROM 编程模式期间，MSP-EXP430F5529LP 上的按钮 S1 和 S2 以及 LED2 用于向用户提供反馈。选择非默认 EEPROM 编程后，eeprom_data.h 和 eeprom_data.c 文件用于对 EEPROM 进行编程。这些文件可由节 2 中提到的 EEPROM 配置工具自动生成。如需更多有关 EEPROM 编程的信息，请参阅节 4.4。

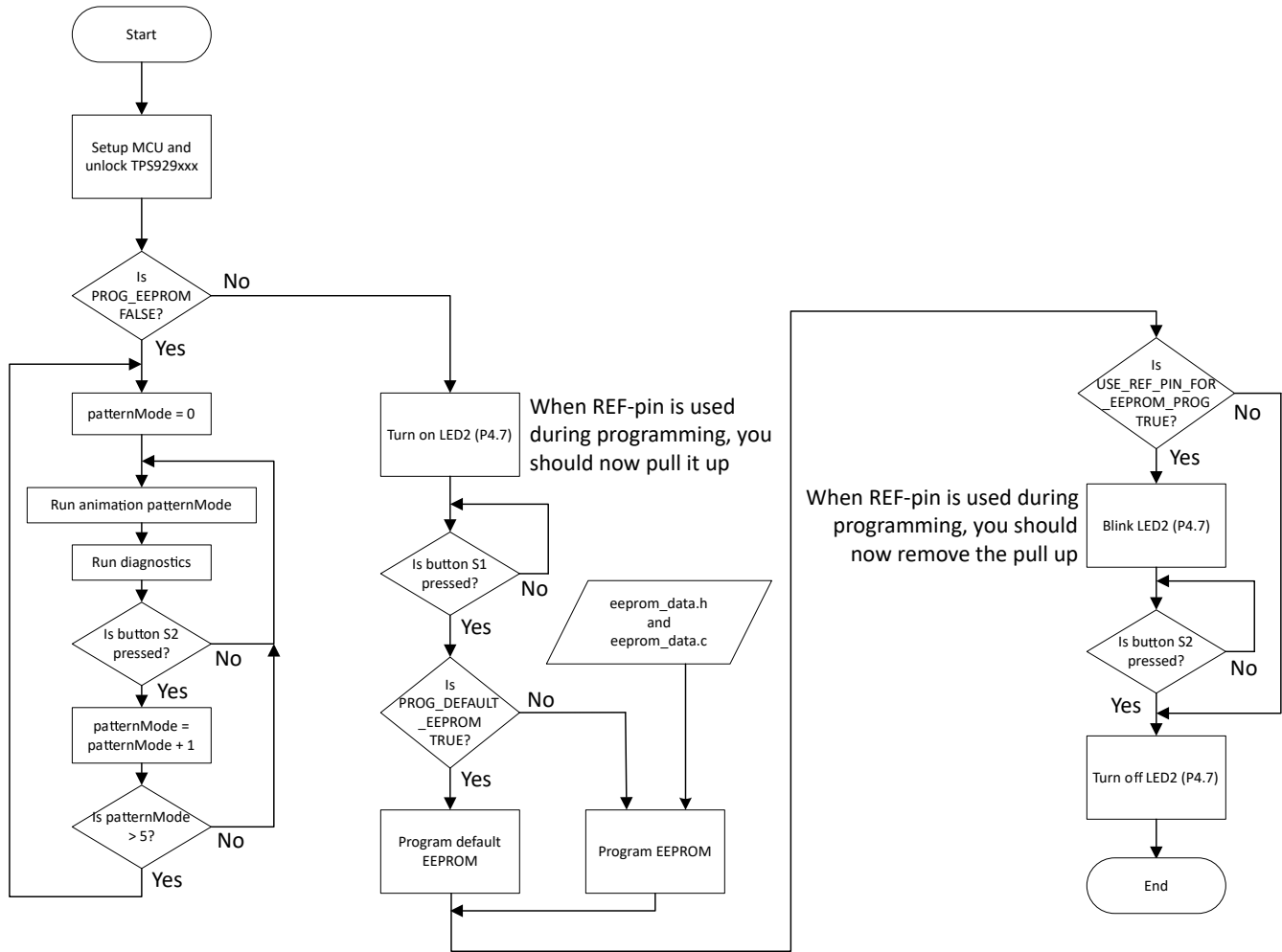


图 4-1. 示例代码流程图

4.2 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 "TPS929240.h"
```

该代码支持：

- TPS929120
- TPS929120A
- TPS929121
- TPS929121A
- TPS929160
- TPS929240
- TPS929240A

请注意，对于“Q1”器件，不会添加此名称，仅使用基本产品名称。所选器件对于处理寄存器中不同的寄存器地址和字段非常重要。此外，在对默认 EEPROM 进行编程时，指定的 LED 驱动器 IC 是用于对默认值进行编程的驱动器 IC。这意味着当用户想要将 TPS929120 编程为 TPS929120A 时，必须在 led_driver.h 文件中选择 TPS929120A。

表 4-1 中列出了影响系统设置及其位置的宏和变量的摘要。

表 4-1. 每个文件的宏和变量名称的摘要

Filename	宏/变量名称	说明
system_info.h	<i>DEVICE_CNT</i>	FlexWire 总线上的器件数量
	<i>CAN_USED</i>	在 UART 或 UART 转 CAN 之间进行选择
	<i>ALWAYS_CHECK_CRC</i>	为所有非广播命令启用 CRC 检查功能
	<i>PROG_EEPROM</i>	启用 EEPROM 编程模式
	<i>PROG_DEFAULT_EEPROM</i>	对默认 EEPROM 值进行编程，而不是对自定义 EEPROM 值进行编程
	<i>USE_REF_PIN_FOR_EEPROM_PROG</i>	在 EEPROM 编程期间使用 REF 引脚
system_info.c	<i>device_address</i>	FlexWire 总线上的器件地址列表
FlexWire.c	<i>rcvCrcError</i>	如果接收到的 CRC 有错误则报告

在 system_info.h 文件中，FlexWire 总线上的器件数由宏 *DEVICE_CNT* 定义。示例代码仅支持 1 条 FlexWire 总线。

```
// Total devices on FlexWire bus
#define DEVICE_CNT 1
```

这些器件的实际地址在 system_info.c 文件中指定。地址序列决定了 FlexWire 非广播写和读命令的顺序。因此，对于不同的器件地址序列，动画模式下的 LED 图形看起来会有所不同。

```
const uint16_t device_address[DEVICE_CNT] = {DEVICE_ADDR_1};
```

system_info.h 文件还定义了其他系统参数。

```
// Define if CAN or UART is used
#define CAN_USED FALSE

// When non-broadcast is transmitted, does the CRC need to be checked
#define ALWAYS_CHECK_CRC FALSE
```

宏 *CAN_USED* 定义是否为 FlexWire 总线使用 UART 或 UART 转 CAN。这会影响在 MCU UART-RX 引脚上接收到的总字节数。

宏 *ALWAYS_CHECK_CRC* 定义对于接收到的反馈，是否每个非广播写入命令都需要检查 CRC。当检查 CRC 后发现其不正确时，全局变量 *rcvCrcError* 设置为 TRUE。在所有其他情况下，该变量设置为 FALSE。变量 *rcvCrcError* 在文件 FlexWire.c 中定义。

```
// When an error in CRC of the received data is observed, set this to TRUE
unsigned int rcvCrcError;
```

4.3 诊断

示例代码提供了一个 API 来检测哪些器件存在开路、短路或单 LED 短路等故障。TPS929xxx_APIs.h 文件中定义了该 API 的原型。

```
void LED_Update_Chip_Status(unsigned int dev_addr_x);
```

该 API 会更新 system_info.h 中定义的变量 *chip_status*。对于器件 TPS929160-Q1 和 TPS929240-Q1，还有一个称为 VBAT 的额外电源引脚。因此，对于这些器件，该变量包括为此引脚测得的电压结果。此外，这些器件还包括一个称为电源欠压的额外故障类型。因此，这些器件包含标志 SUPUV。

```

struct chipStatus {
    // Indicates open, short, and/or single-LED-short fault
    uint16_t OUT_flag;
    uint16_t SHORT_channels[MAX_CHANNEL_CNT];
    uint16_t OPEN_channels[MAX_CHANNEL_CNT];
    uint16_t SLS_channels[MAX_CHANNEL_CNT];           // Single-LED-short
    uint16_t EEPROM_CRC;                             // EEPROM CRC fault
    uint16_t TSD;                                     // Thermal Shutdown
    uint16_t PRETSD;                                  // Pre-thermal shutdown warning
    uint16_t REF;                                     // REF-pin fault
    uint16_t LOWSUP;                                  // Low supply
    uint16_t POR;                                     // Power-on-reset
#ifndef TPS92912X
    uint16_t SUPUV;                                   // Supply undervoltage
    uint16_t VBAT_mV;                                // *1 mV
#endif
    uint16_t VSUPPLY_mV;                              // *1 mV
    uint16_t VLDO_mV;                                 // *1 mV
    uint16_t TEMPSNS_10mC;                            // *10 mC
    uint16_t VREF_100uV;                              // *100 uV
    uint16_t IREF_10nA;                               // *10 nA
};

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

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 4-2 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 FlexWire 总线上 LED 驱动器的地址。总共可能有 16 个不同的地址。因此，索引的范围为 0 至 15。

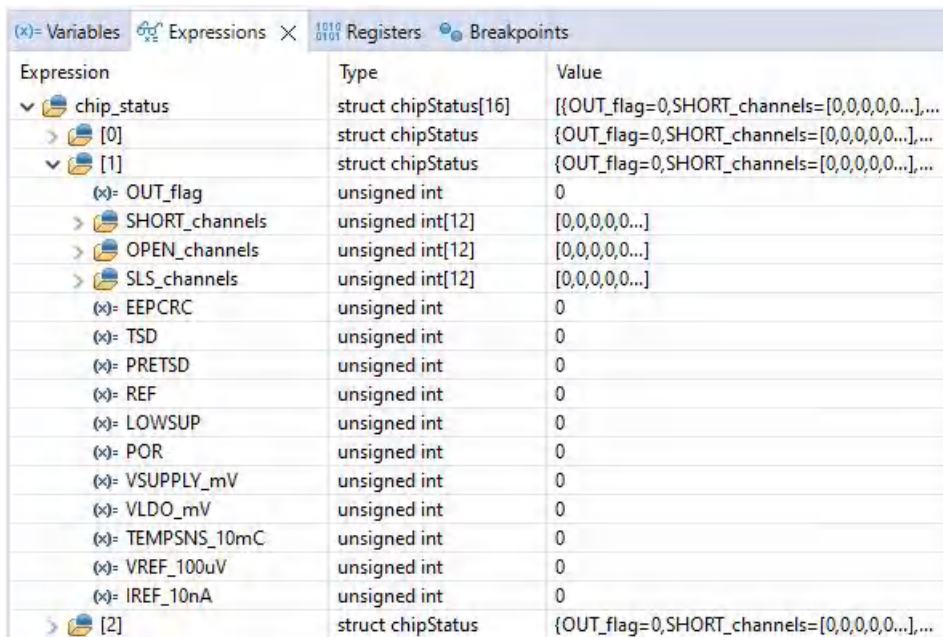
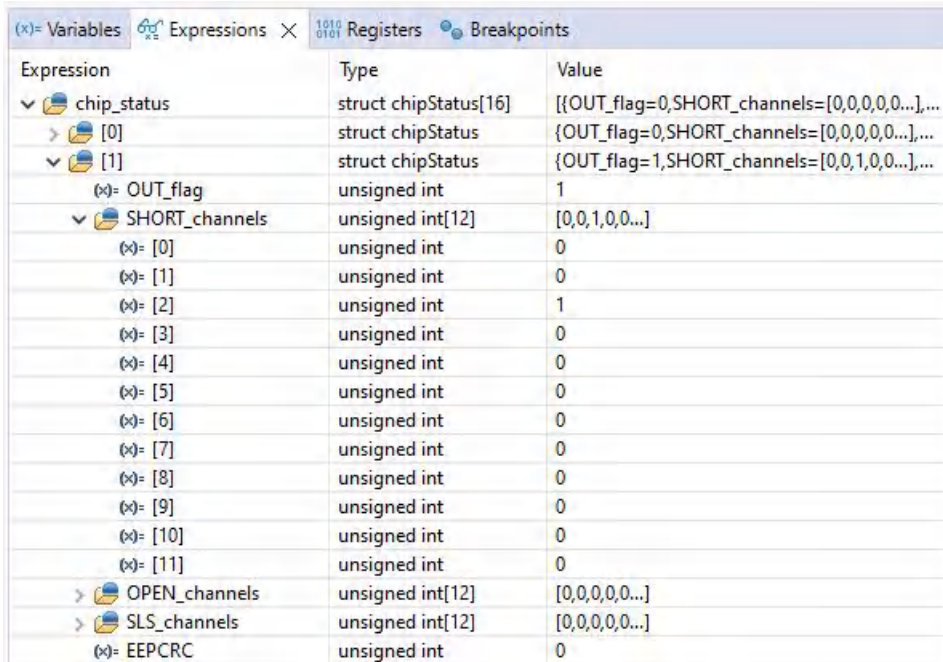


图 4-2. 观察 TPS929120-Q1 的表达式 `chip_status` (无错误) 的示例

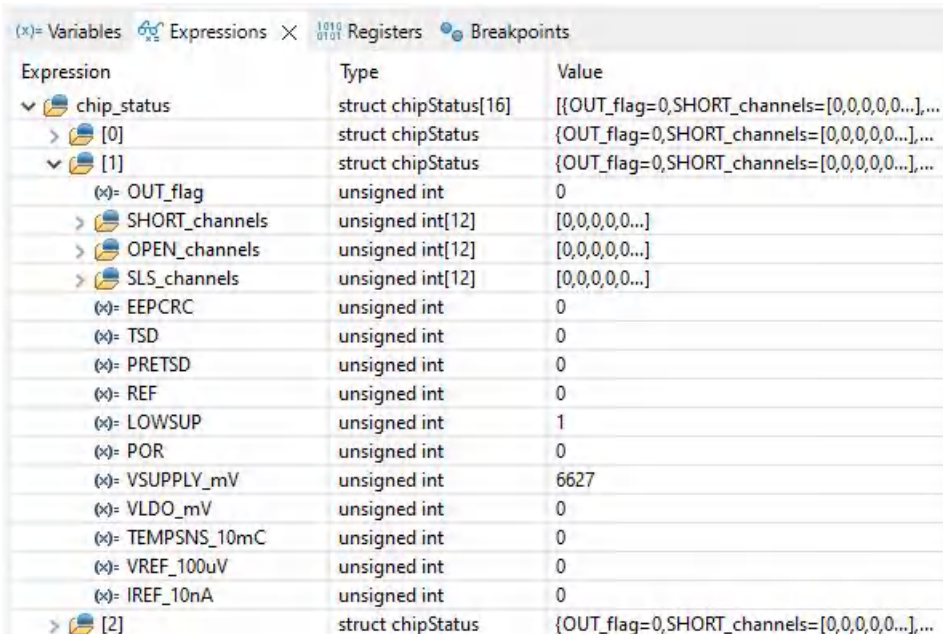
图 4-3 中显示了一个带有短接的示例。TPS929120-Q1 具有地址 0x1 并设置了 OUT_Flag 标志。当阵列 SHORT_channels 被扩展时，可以看到引脚 OUT2 上发生了短路。



Expression	Type	Value
chip_status	struct chipStatus[16]	{{OUT_flag=0,SHORT_channels=[0,0,0,0,0...],...
[0]	struct chipStatus	{OUT_flag=0,SHORT_channels=[0,0,0,0,0...],...
[1]	struct chipStatus	{OUT_flag=1,SHORT_channels=[0,0,1,0,0...],...
OUT_flag	unsigned int	1
SHORT_channels	unsigned int[12]	[0,0,1,0,0...]
[0]	unsigned int	0
[1]	unsigned int	0
[2]	unsigned int	1
[3]	unsigned int	0
[4]	unsigned int	0
[5]	unsigned int	0
[6]	unsigned int	0
[7]	unsigned int	0
[8]	unsigned int	0
[9]	unsigned int	0
[10]	unsigned int	0
[11]	unsigned int	0
OPEN_channels	unsigned int[12]	[0,0,0,0,0...]
SLS_channels	unsigned int[12]	[0,0,0,0,0...]
EEPCRC	unsigned int	0

图 4-3. 观察 TPS929120-Q1 的表达式 chip_status (具有短路故障) 的示例

图 4-4 中展示了 TPS929120-Q1 中出现低电源警告 ($V_{\text{SUPPLY}} < V_{\text{ADCLOWSUPTH}}$) 时的示例。已为地址为 0x1 的器件设置了标志 LOWSUP。此外，对于该警告，电源电压由 ADC 测量并在诊断中报告。在本例中，测量得到的结果为 6627mV。



Expression	Type	Value
chip_status	struct chipStatus[16]	{{OUT_flag=0,SHORT_channels=[0,0,0,0,0...],...
[0]	struct chipStatus	{OUT_flag=0,SHORT_channels=[0,0,0,0,0...],...
[1]	struct chipStatus	{OUT_flag=0,SHORT_channels=[0,0,0,0,0...],...
OUT_flag	unsigned int	0
SHORT_channels	unsigned int[12]	[0,0,0,0,0...]
OPEN_channels	unsigned int[12]	[0,0,0,0,0...]
SLS_channels	unsigned int[12]	[0,0,0,0,0...]
EEPCRC	unsigned int	0
TSD	unsigned int	0
PRETSD	unsigned int	0
REF	unsigned int	0
LOWSUP	unsigned int	1
POR	unsigned int	0
VSUPPLY_mV	unsigned int	6627
VLDO_mV	unsigned int	0
TEMPSNS_10mC	unsigned int	0
VREF_100uV	unsigned int	0
IREF_10nA	unsigned int	0
[2]	struct chipStatus	{OUT_flag=0,SHORT_channels=[0,0,0,0,0...],...

图 4-4. 观察 TPS929120-Q1 的表达式 chip_status (具有低电源) 的示例

4.4 EEPROM 编程

示例代码包括对 EEPROM 进行编程的功能。此功能由 system_info.h 文件中定义的宏启用。

```
// When set to 1, the EEPROM programming routine is executed instead of normal program
#define PROG_EEPROM (FALSE)
// When set to 1, program the EEPROM to the default value
#define PROG_DEFAULT_EEPROM (FALSE)
// Use external device address settings for EEPROM programming
#define USE_REF_PIN_FOR_EEPROM_PROG (FALSE)
```

当宏 `PROG_EEPROM` 被定义为 TRUE 时，EEPROM 编程模式被启用。示例代码可以对指定 LED 驱动器 IC 或自定义设置的默认 EEPROM 值进行编程。当 `PROG_DEFAULT_EEPROM` 宏定义为 FALSE 时，会对自定义设置进行编程。此设置在 eeprom_data.h 和 eeprom_data.c 文件中指定。这些文件可由节 2 中提到的 EEPROM 配置工具自动生成。

LED 驱动器 IC 支持两种针对单独芯片选择的解决方案，通过拉高 REF 引脚或通过按地址引脚配置器件地址来实现。当 `USE_REF_PIN_FOR_EEPROM_PROG` 宏被定义为 TRUE 时，REF 引脚应在被编程期间被拉高。当该宏被定义为 FALSE 时，使用当前器件地址。TI 建议使用当前器件地址。

当代码进入 EEPROM 编程例程时，它会使 MSP-EXP430F5529LP 上的 LED2 (P4.7) 亮起。当 `USE_REF_PIN_FOR_EEPROM_PROG` 宏定义为 TRUE 时，REF 引脚应该在 LED2 亮起后被上拉。表 4-2 中列出了为每个 EVM 上拉 REF 引脚所需的跳线。

LED2 亮起后，应按下 MSP-EXP430F5529LP 上的按钮 S1 以开始编程。当使用当前器件地址时，LED2 将在编程完毕后熄灭。

当使用 REF 引脚时，LED2 会在编程完毕后开始闪烁。此时，REF 引脚上的上拉电阻应被移除，然后应按下 MSP-EXP430F5529LP 上的按钮 S2。然后，LED2 将熄灭。

表 4-2. 在 EEPROM 编程期间使用 REF 引脚时要设置的 EVM 跳线

EVM	跳线
TPS929120EVM	J2 位置 2 和 3 (+5V)
TPS929160EVM	J52 位置 2 和 3 (VLDO)
TPS929240EVM	J10 位置 2 和 3 (VLDO)

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

Copyright © 2023，德州仪器 (TI) 公司