User's Guide

LP581X RUKA 示例代码

摘要

本文档介绍了将 LP5815 器件与 LP-MSPM0L1306 器件配对使用时，示例代码的准备与使用方法。按照德州仪器 (TI) 提供的安装指南进行操作后，安装好的代码将点亮 LP5815YCHEVM 评估模块上的发光二极管 (LED)。

LP5815 是一款具有即时闪烁和自主动画控制功能的三通道集成电路内置总线 (I2C) 接口红、绿、蓝 (RGB) LED 驱动芯片。您可以修改示例代码以控制 LP5814、LP5814I、LP5816 和 LP5817 器件系列。

内容

1 简介.....2
2 硬件设置.....3
3 软件设置.....4
4 示例代码结构.....6
4.1 设计要求.....6
4.2 流程图.....6
4.3 模式参数.....8
5 修订历史记录.....11

插图清单

图 2-1. 硬件连接.....3
图 3-1. Code Composer Studio 安装过程.....4
图 3-2. MSPM0 SDK 安装验证.....5
图 4-1. 示例代码流程图.....8
图 4-2. 模式示例 1 时序和参数.....9
图 4-3. 模式示例 2 时序和参数.....9
图 4-4. 模式示例 3 时序和参数.....11

表格清单

表 4-1. LP5815 设计参数.....6
表 4-2. 引擎繁忙锁定寄存器.....7

商标

1 简介

该示例代码演示了如何通过 I2C 通信，使用 LP-MSPM0L1306 评估板配置 LP5815 器件，以点亮 LP5815YCHEVM 评估模块上的 RGB LED。

代码中演示了三个模式示例：

- 手动模式下 LED 的亮起和熄灭
- 手动模式下 LED 的淡入淡出（呼吸灯）
- 自主动画模式下 LED 由呼吸灯效果切换为闪烁效果

所有三个模式示例都可以在 LP5814、LP5814I 和 LP5815 上实现。对于 LP5816 和 LP5817，仅支持前两种模式示例。

2 硬件设置

本节介绍了如何在 LP5815YCHEVM 评估模块与 LP-MSPM0L1306 评估板之间建立连接。开始评估时，需要使用以下列表中的各项内容：

- 计算机
- LP5815YCHEVM 评估模块
- LP-MSPM0L1306 评估板

在 LP-MSPM0L1306 评估板上保留默认跳线设置。LP5815YCHEVM 评估模块可以直接由 LP-MSPM0L1306 评估板的 3.3V 电源供电，不需要外接电源。图 2-1 中演示了电源和 I2C 连接。设置过程如下文所示：

1. 取下 LP5815YCHEVM 评估模块上 J3 接头上的跳线。保留 J2 和 J9 接头上的跳线，使 LP5815 器件的公共集电极电源 (VCC) 与 LED 电源 (VLED) 连接到同一 3.3V 电源轨。
2. 请参阅图 2-1，将 LP5815YCHEVM 评估模块的 GND、SCL、SDA、STAT 和 3.3V 接口连接至 LP-MSPM0L1306 评估板。
3. 通过套件随附的 USB 电缆将 LP-MSPM0L1306 评估板连接到计算机。

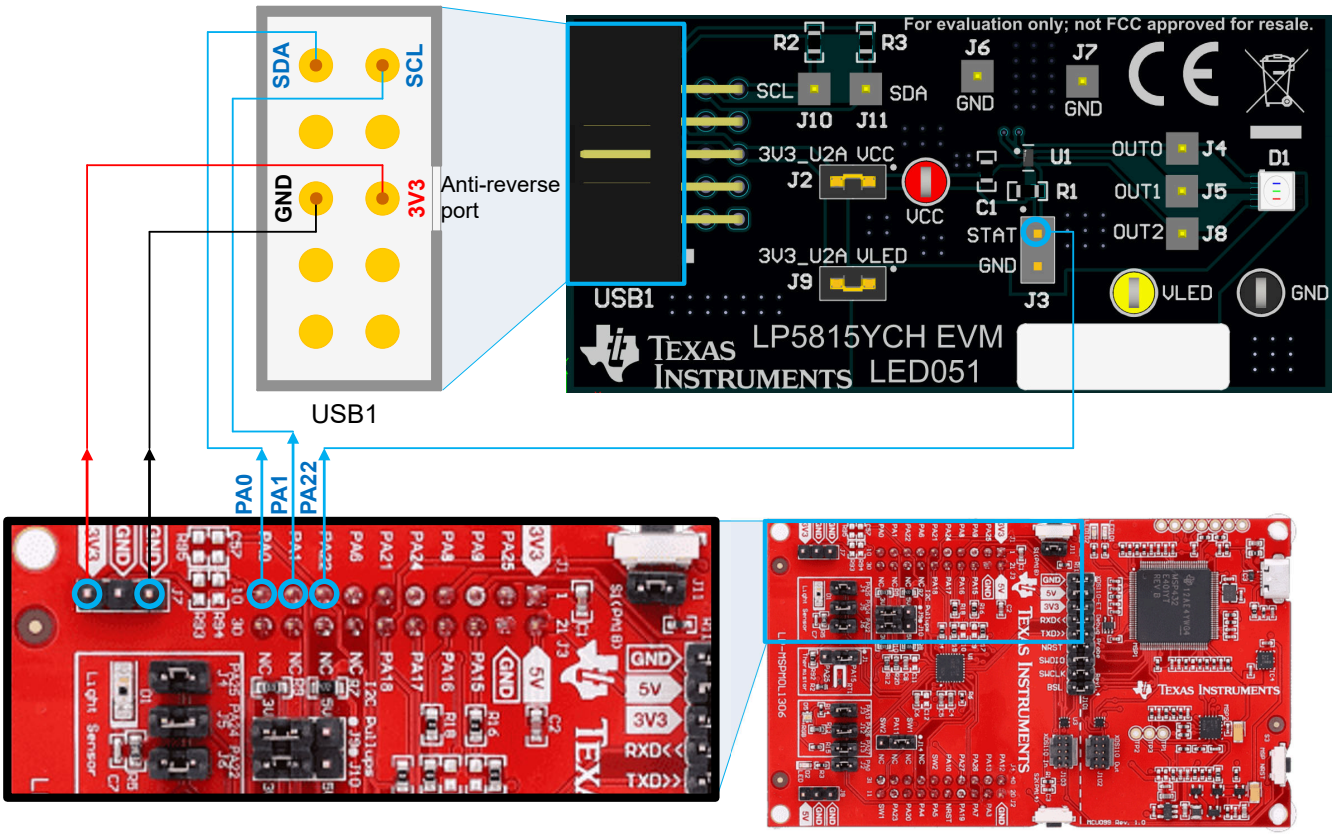

图 2-1. 硬件连接

3 软件设置

要为 LP-MSPM0L1306 LaunchPad™ 设置软件，请执行以下步骤（在装有 Windows 11 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 ≥ 12.5.0）。
   b. 按照安装说明安装 Code Composer Studio。在安装过程中，如果您将 *Setup type* 选择为 *Custom Installation*，请在 *Select Components* 中勾选 **MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs**，如图 3-1 中红框所示。

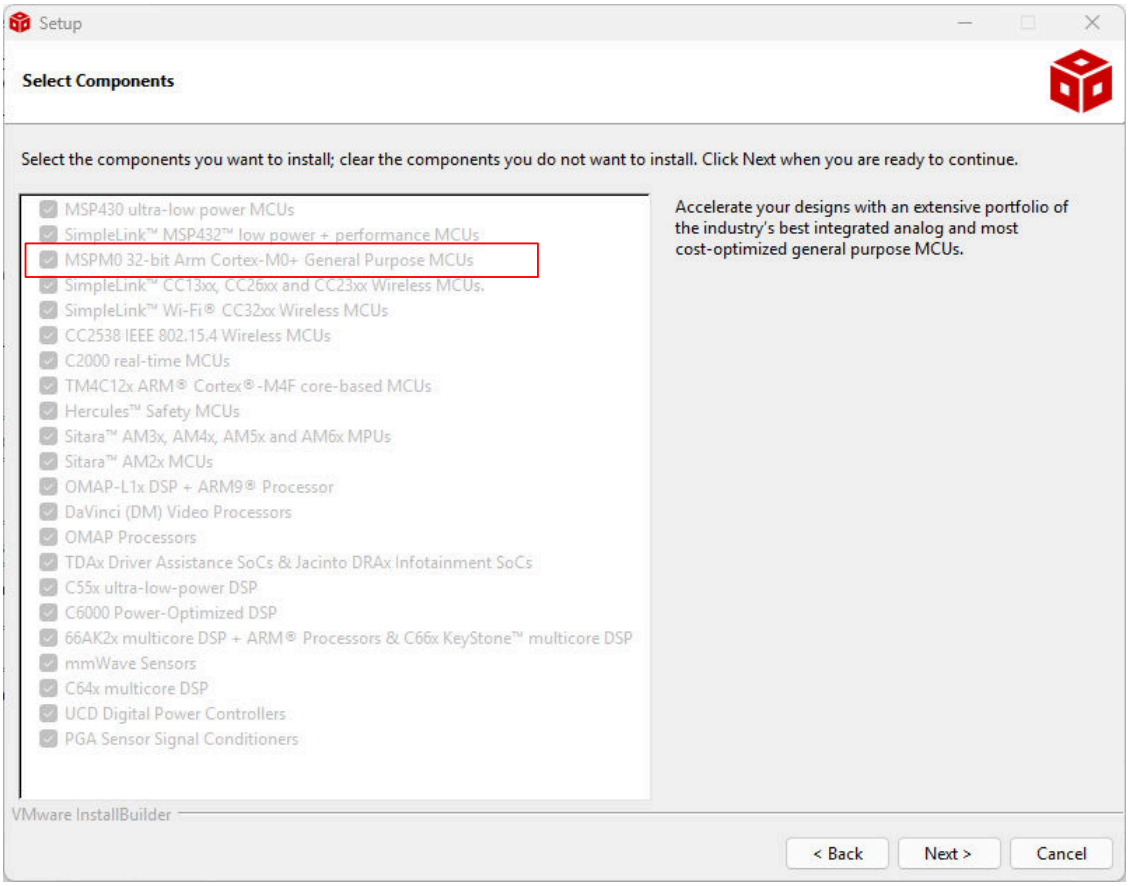

图 3-1. Code Composer Studio 安装过程

2. 下载并安装 MSPM0-SDK（版本 ≥ 2.4.xx.xx）。
   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*。然后从下拉列表中选择 *Preferences*。将显示 *Preferences* 窗口。从左侧栏中，依次选择 *Code Composer Studio* → *Products*。
   b. 请确保在 *Discovered products* 下方显示有 *MSPM0 SDK* 和 *SysConfig*（SysConfig 会在您安装 **MSPM0 32-bit Arm Cortex-M0+ General Purpose MCUs** 时自动安装），如图 3-2 中所示。如果未显示 *MSPM0 SDK* 或 *SysConfig*，请选择 *Preference* 窗口右侧的 *Refresh*。如果 *MSPM0 SDK* 或 *SysConfig* 仍然不可用，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。

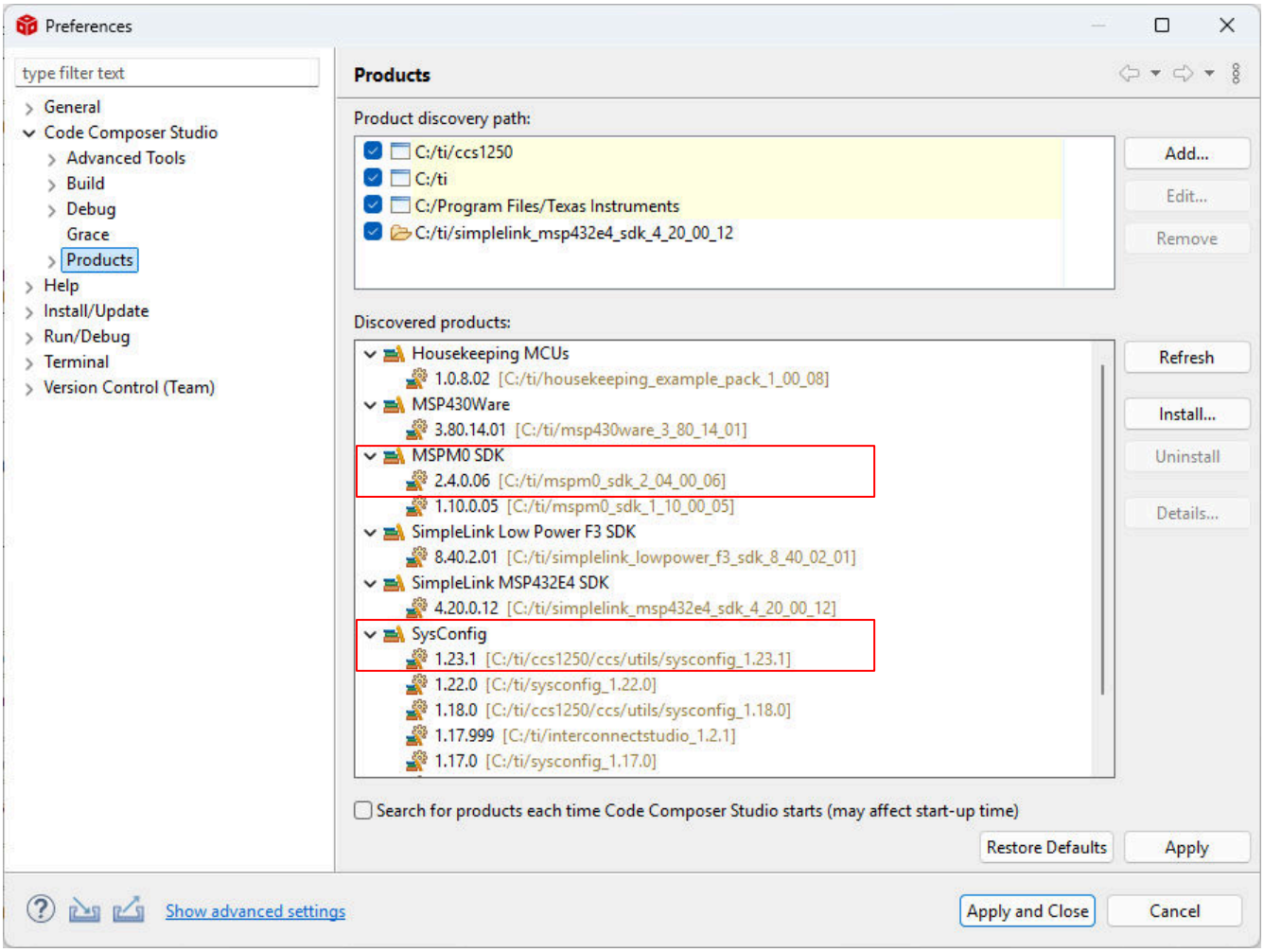

图 3-2. MSPM0 SDK 安装验证

3. 下载并导入示例代码。将压缩的示例代码文件下载到本地计算机后，根据链接中提供的过程提取文件并导入 Code Composer Studio (CCS) 工程：导入 CCS 工程。
4. 根据链接中提供的过程将程序加载至 MCU：构建和运行工程。

4 示例代码结构

4.1 设计要求

表 4-1 中列出了 LP5815 器件的设计参数。

表 4-1. LP5815 设计参数

设计参数	LP5815
VCC 和 VLED	3.3V
OUT0 最大电流（蓝色 LED）	8mA
OUT1 最大电流（绿色 LED）	14mA
OUT2 最大电流（红色 LED）	20mA
I2C 频率	400kHz
I2C 目标地址	0x2D

I2C 目标地址在文件 lp5815_example.h 中定义为宏。支持两种地址模式：独立地址模式和广播地址模式。

1. 独立地址 (*I2C_TARGET_ADDRESS_INDEPENDENT*)：当与不同的设备进行通信时，该地址是不同的。
   a. 0x2D 用于与 LP5815 和 LP5817 通信。
   b. 0x2C 用于与 LP5814 和 LP5816 通信。
   c. 0x2E 用于与 LP5814I 通信。
2. 广播地址 (*I2C_TARGET_ADDRESS_BROADCAST*)：当与所有设备进行通信时，该地址是相同的。该地址为 0x34。

```
#define I2C_TARGET_ADDRESS_INDEPENDENT (0x2D);  //b0010 1101 - LP5815, LP5817

#define I2C_TARGET_ADDRESS_BROADCAST (0x34);  //b0011 0100
```

4.2 流程图

图 4-1 描述了示例代码中的高级流程。

这三个模式示例被放置在 while 循环中。有关三个模式示例的详细说明，请参阅模式参数部分。lp5815_example_patterns.h 文件中记录了相关代码。在执行模式示例 1 和模式示例 2 时，LP5815 器件被配置为手动模式。在执行模式示例 3 时，LP5815 器件被配置为自主动画模式。

必须执行发送 *Update_command* 的步骤，才能使器件配置寄存器值生效。**将器件配置寄存器从 *DEV_CONFIG0* 修改为 *DEV_CONFIG4* 后，请发送 *Update_command*。**

```
/* send update cmd to make device configuration registers value take effect */
startRegAddress = UPDATE_CMD_REG;
tempData[0] = UPDATE_CMD_VAL; //data to write
i2cSendReceive(I2C_TARGET_ADDRESS_INDEPENDENT, startRegAddress, i2cWrite, 1, &tempData[0]);
```

当 *FLAG* 寄存器中的 *ENGINE_BUSY* 标志置为 1 时，表 4-2 中所示的引擎配置寄存器和模式配置寄存器会被锁定，以防止被修改。只有在 **ENGINE_BUSY = 0** 时，才能修改这些引擎繁忙锁定寄存器。

```
/*
 * need to check ENGINE_BUSY flag before,
 * engine configure registers, from ENGINE_CONFIG0 to ENGINE_CONFIGURE6,
 * and all pattern configure registers can be configured
 * read ENGINE_BUSY bit in FLAG register, the read back data stored in gRxPacket[]
 */
startRegAddress = FLAG_REG;
i2cSendReceive(I2C_TARGET_ADDRESS_INDEPENDENT, startRegAddress, i2cRead, 1, &tempData[0]);
/* send stop cmd to stop animation if ENGINE_BUSY = 1 */
if(gRxPacket[0] & 0x04)
{
```

```
        startRegAddress = STOP_CMD_REG;
        tempData[0] = STOP_CMD_VAL; //data to write
        i2cSendReceive(I2C_TARGET_ADDRESS_INDEPENDENT, startRegAddress, i2cWrite, 1, &tempData[0]);
    }
```

表 4-2. 引擎繁忙锁定寄存器

说明	寄存器地址	寄存器缩写
引擎配置寄存器	0x06 至 0x0C	ENGINE_CONFIG0 至 ENGINE_CONFIG6
模式配置寄存器	0x1C 至 0x3F	• PATTERN*x*_PAUSE_TIME • PATTERN*x*_REPEAT_TIME • PATTERN*x*_PWM0 • PATTERN*x*_PWM1 • PATTERN*x*_PWM2 • PATTERN*x*_PWM3 • PATTERN*x*_PWM4 • PATTERN*x*_SLOPER_TIME1 • PATTERN*x*_SLOPER_TIME2 *x* = 0、1、2、3

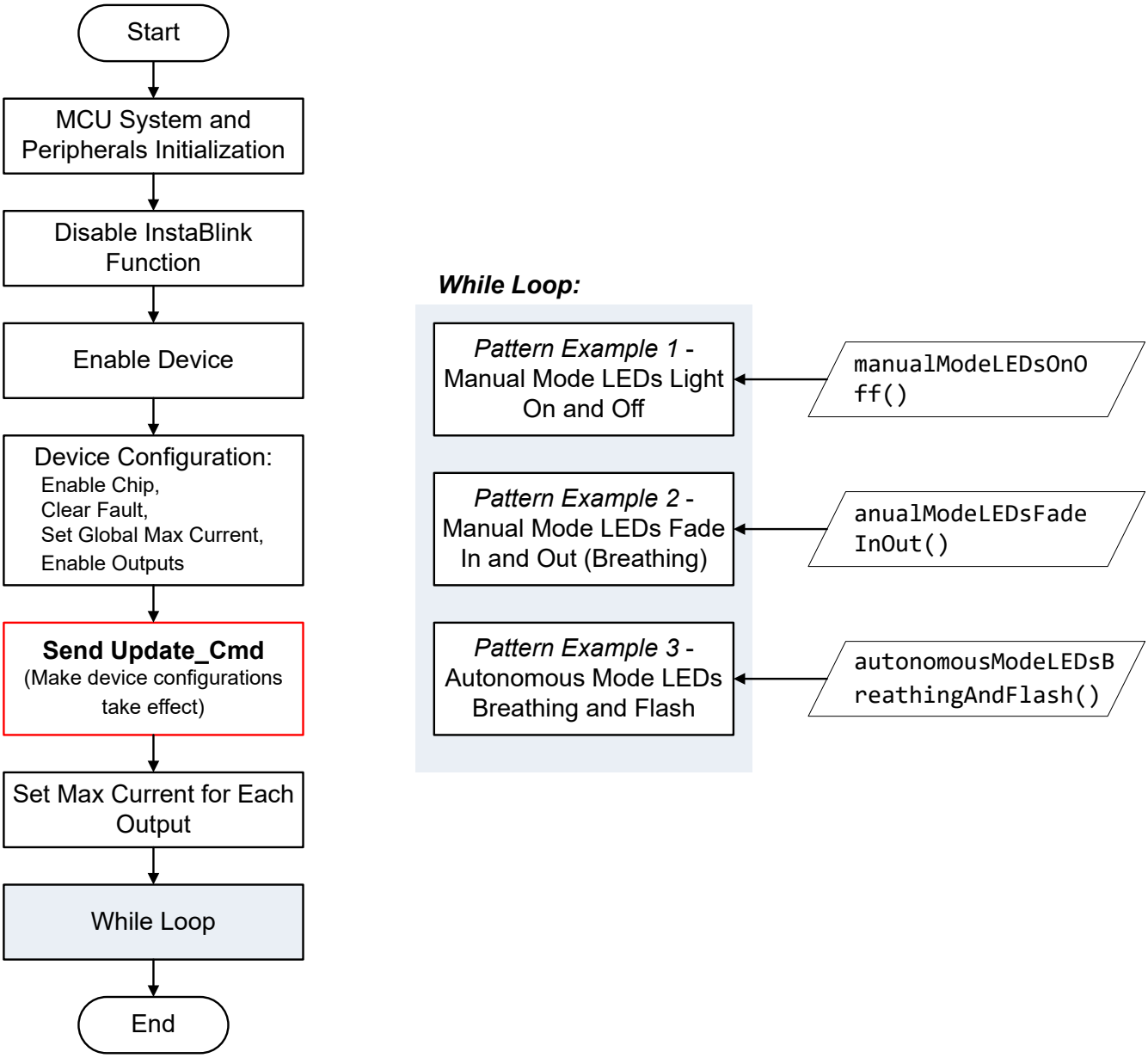

图 4-1. 示例代码流程图

4.3 模式参数

模式示例 1

相应函数的示例为 manualModeLEDsOnOff()。在此示例中，LP5815 被配置为手动模式。通过周期性地向 *OUTx_MANUAL_PWM (x = 0、1、2)* 寄存器写入 0x80 和 0x00，可同时控制红、绿、蓝 LED 的点亮与熄灭。如图 4-2 所示，通过系统延时函数 delay_ms() 实现点亮和熄灭各持续 500ms。

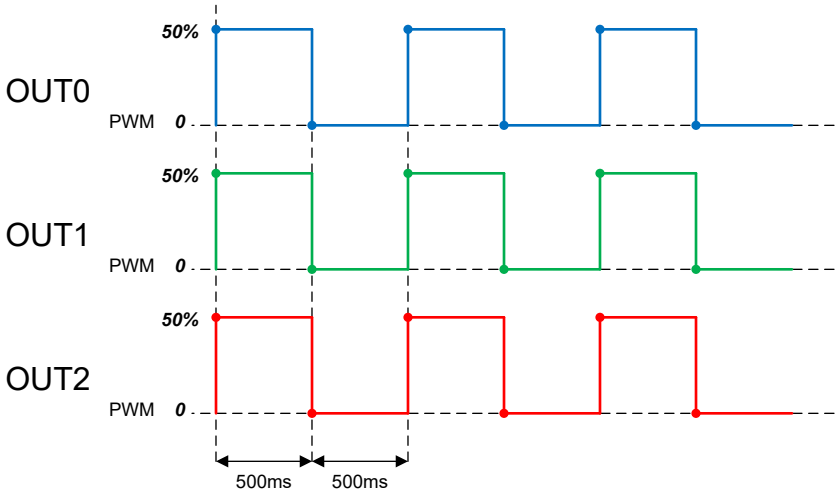

图 4-2. 模式示例 1 时序和参数

模式示例 2

相应函数的示例为 manualModeLEDsFadeInOut()。在此示例中，LP5815 被配置为手动模式。

为了实现渐亮和渐灭功能，通过将 *DEV_CONFIG2* 寄存器中的 *OUTx_FADE_EN (x = 0、1、2)* 位置位，启用 LP5815 的 Sloper 功能。各输出可自动从当前 PWM 值平滑过渡到更新后的 PWM 值，实现 256 级渐亮或渐灭效果，从而减少 MCU 的通信负载。为了获得更加符合人眼视觉感受的调光效果，通过将 *DEV_CONFIG3* 寄存器中的 *OUTx_EXP_EN (x = 0、1、2)* 位置位，启用指数调光模式。如图 4-3 所示，通过将 *DEV_CONFIG2* 寄存器中的 *LED_FADE_TIME* 位设置为 0xB，可实现 1s 淡入和淡出时间。

确保发送 *Update_command*，使器件配置寄存器值生效。

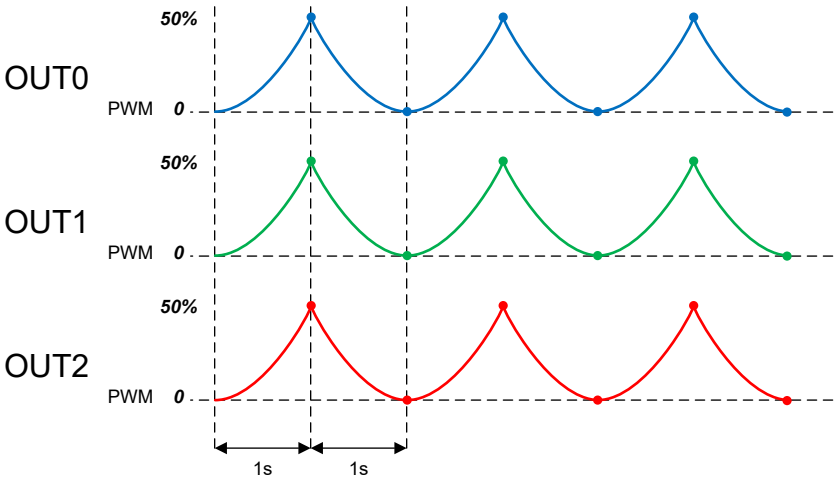

图 4-3. 模式示例 2 时序和参数

模式示例 3

相应函数的示例为 autonomousModeLEDsBreathingAndFlash()。在此示例中，通过设置 *DEV_CONFIG3* 寄存器中的 *OUTx_AUTO_EN (x = 0、1、2)* 位，将 LP5815 配置为自主动画模式。

通过设置 *DEV_CONFIG4* 寄存器中的 *OUTx_ENGINE_CH (x = 0、1、2)* 位，OUT0 选择 ENGINE0，OUT1 选择 ENGINE1，OUT2 选择 ENGINE2。相应的代码如下所示。

确保发送 *Update_command*，使器件配置寄存器值生效。

```
/*
 * OUT2 select ENGINE 2
 * OUT1 select ENGINE 1
 * OUT0 select ENGINE 0
 */
startRegAddress = DEV_CONFIG4;
tempData[0] = 0x24; //data to write
i2cSendReceive(I2C_TARGET_ADDRESS_INDEPENDENT, startRegAddress, i2cWrite, 1, &tempData[0]);
```

请先确认 *ENGINE_BUSY* 标志为 0，然后再设置 *ENGINE_CONFIG0* 至 *ENGINE_CONFIG6* 引擎配置寄存器。

启用引擎顺序并为引擎顺序选择模式，以便为 OUT0、OUT1 和 OUT2 构建模式。通过设置 *ENGINE_CONFGIG4* 和 *ENGINE_CONFGIG5* 寄存器中的 *ExOy_EN (x = 0、1、2 ; y = 0、1)* 位，为每个引擎启用两个引擎顺序。相应的代码如下所示。

```
/*
 * enable ENGINE0_ORDER0 and ENGINE0_ORDER1,
 * enable ENGINE1_ORDER0 and ENGINE1_ORDER1,
 * enable ENGINE2_ORDER0 and ENGINE2_ORDER1
 */
startRegAddress = ENGINE_CONFIG4;
tempData[0] = 0x33; //data to write
tempData[1] = 0x03;
i2cSendReceive(I2C_TARGET_ADDRESS_INDEPENDENT, startRegAddress, i2cWrite, 2, &tempData[0]);
```

通过设置 *ENGINE_CONFGIG0* 至 *ENGINE_CONFGIG2* 寄存器中的 *ENGINEx_ORDERy (x = 0、1、2 ; y = 0、1)* 位，为各引擎顺序选择对应的模式，从而构建 OUT0、OUT1 和 OUT2 的最终模式。相应的代码如下所示。

```
/*
 * ENGINE0: ENGINE0_ORDER0 select PATTERN0, ENGINE0_ORDER1 select PATTERN3, (0x0C)
 * ENGINE1: ENGINE1_ORDER0 select PATTERN1, ENGINE1_ORDER1 select PATTERN3, (0x0D)
 * ENGINE2: ENGINE2_ORDER0 select PATTERN2, ENGINE2_ORDER1 select PATTERN3, (0x0E)
 */
startRegAddress = ENGINE_CONFIG0;
tempData[0] = 0x0C; //data to write
tempData[1] = 0x0D;
tempData[2] = 0x0E;
i2cSendReceive(I2C_TARGET_ADDRESS_INDEPENDENT, startRegAddress, i2cWrite, 3, &tempData[0]);
```

通过设置 *ENGINE_CONFGIG6* 寄存器中的 *ENGINEx_REPT (x = 0、1、2)* 位，可使整个引擎模式重复两次。

LP5815 具有 4 个独立的可配置模式单元，即 *PATTERN0* 至 *PATTERN3*。在本例中，PATTERN0、PATTERN1、PATTERN2 和 PATTERN3 全部已使用。有关每个模式单元的模式参数设置，请参阅图 4-4。lp5815_example_patterns.h 文件中的 setPatternParameters() 函数用于配置每个模式单元的所有参数，如下所示。

请先确认 *ENGINE_BUSY* 标志为 0，然后再设置 *PATTERN0_PAUSE_TIME* 至 *PATTERN3_SLOPER_TIME2* 模式参数配置寄存器。

```
/*
 *  @brief      set pattern parameters
 *
 *  @param[in]  patternBaseAdr  i2c slave address
 *  @param[in]  pauseT0  start register address to write or read
 *  @param[in]  pauseT1  write or read operation
 *  @param[in]  pt data length of the data to write or read
 *  @param[in]  pwm0/pwm1/pwm2/pwm3/pwm4  point to the data array to write
 *  @param[in]  sloperT0/sloperT1/sloperT2/sloperT3
 */
void setPatternParameters(uint8_t patternBaseAdr, uint8_t pauseT0, uint8_t pauseT1, uint8_t pt,
                          uint8_t pwm0, uint8_t pwm1, uint8_t pwm2, uint8_t pwm3, uint8_t pwm4,
                          uint8_t sloperT0, uint8_t sloperT1, uint8_t sloperT2, uint8_t sloperT3)
{
    uint8_t startRegAddress; //the start address to write or read from

    startRegAddress = patternBaseAdr;
```

```
    tempData[0] = (pauseT0 << 4) | (pauseT1); //data to write
    tempData[1] = pt;
    tempData[2] = pwm0;
    tempData[3] = pwm1;
    tempData[4] = pwm2;
    tempData[5] = pwm3;
    tempData[6] = pwm4;
    tempData[7] = (sloperT1 << 4) | (sloperT0);
    tempData[8] = (sloperT3 << 4) | (sloperT2);
    i2cSendReceive(I2C_TARGET_ADDRESS_INDEPENDENT, startRegAddress, i2cWrite, 9, &tempData[0]);
}
```

完成所有模式参数设置后，**发送 *Start_command*，即可启动在输出端运行的自主动画**。*ENGINE_BUSY* 标志设置为 1，表示输出端正在运行自主模式。当模式停止时，*ENGINE_BUSY* 标志会自动清理。

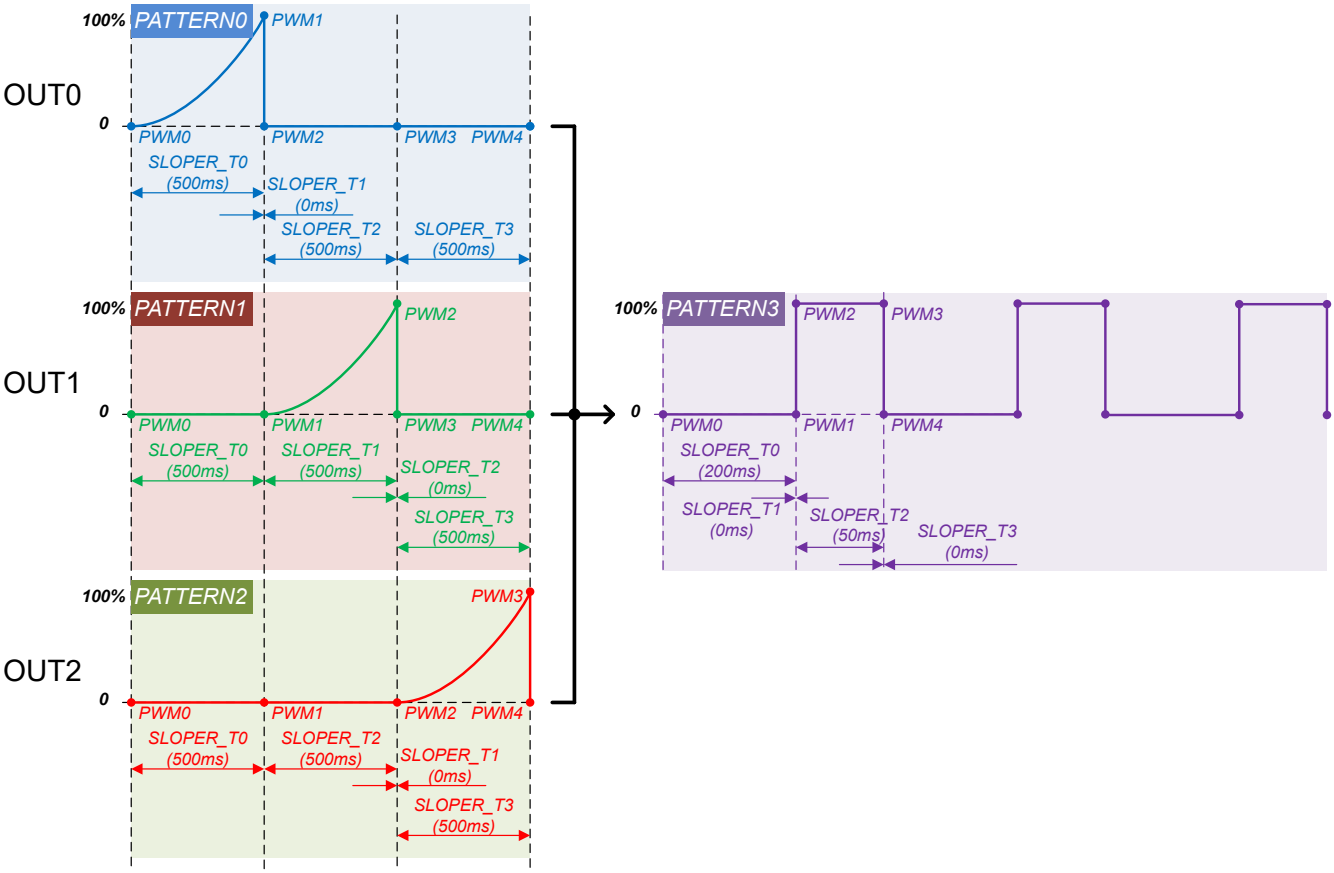

图 4-4. 模式示例 3 时序和参数

5 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

日期	修订版本	注释
July 2026	*	初始发行版

重要通知和免责声明

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、与某特定用途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安全、安保法规或其他要求。

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 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 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。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月