Alienware AlienFX SDK 5.2

用户指南

摘要

此软件开发包 (SDK) 包含的功能与函数可供应用程序开发人员在 AlienFX 硬件平台和/或已连接 AlienFX 兼容设备的平台上配置彩色灯光，以便实现多种所需的视觉效果。

2019 年 3 月

版本

版本	SDK 版本	发布日期	注释
2.0	5.2	2019 年 3 月 6 日	• 阐明了已被弃用管理的 DLL 和函数 • 更改了开发要求以支持最新版本的 SDK • 添加了常见问题解答和故障排除附录
1.2	2.1	2012 年 3 月 26 日	添加了 LFX_GetVersion 函数
1.1	2.0	2011 年 5 月 1 日	添加了 game configurator 函数，
1.0	1.0	2009 年 3 月 24 日	Alpha 版本

目录

1 简介

1.1 识别

Dell Alienware AlienFX 软件开发包 (SDK) 适用于为 AlienFX 硬件平台和/或已连接 AlienFX 兼容设备的平台开发应用程序的用户。

注意：采用 AlienFX SDK 5.2 开发的应用程序向后兼容使用前代 AlienFX SDK 的系统。

1.2 目的

本文旨在提供 Alienware AlienFX SDK 中可用功能与函数的详细编程概述。通过阅读本文，应用程序开发人员能够针对连接到系统的任何可用 Alienware AlienFX 设备实现灯光控制。通过控制 Alienware AlienFX 生态系统，应用程序可能会使用彩色灯对设备进行配置，以响应应用程序请求并实现多种所需的视觉效果。

1.3 范围

本文包含 LightFX.dll 库的可用特性与功能，包括其函数、相关库、数据以及在系统中操控彩色灯所需的方法。

注意：不支持采用 AlienFX SDK 2.1 及更低版本（使用 AWCC 4.x 部署）进行开发。

---

2 使用入门

2.1 SDK 5.2 内容

该软件开发包 (SDK) 由灯光库、应用程序示例和本文组成。安装 Alienware Command Center 后，表 1 中的文件（如下所示）将包含在安装目录的子目录当中（例如，C:\Program Files\Alienware\Command Center）。

Table 1 资料库和位置

资料	位置
说明文件	[InstallDir]\AlienFX SDK\
标头及所含文件	[InstallDir]\AlienFX SDK\includes\
动态链接库	• [InstallDir]\AlienFX SDK\DLLs\x86\LightFX.dll • [InstallDir]\AlienFX SDK\DLLs\x64\LightFX.dll 您还可以在此处找到相同的库： • \Windows\System32\LightFX.dll （64 位版本） • \Windows\SysWOW64\LightFX.dll （32 位版本）
示例	[InstallDir]\AlienFX SDK\Samples\

警告：请勿将上述库分发至使用 Alienware AlienFX SDK 开发的应用程序。应用程序的最终版本必须使用通过 Alienware Command Center 部署的 LightFX.dll。

2.2 开发的系统要求

对于 2018 年及以后发布的 AlienFX 平台及外围设备来说，必须安装 AWCC 5.2 或更高版本才能使用 AlienFX SDK 5.2 进行开发。

Table 2 开发的系统要求

类型	要求
操作系统	AlienFX SDK 5.x：Windows 10，64 位
软件	Alienware Command Center 5.2.x
硬件	Alienware AlienFX 5.2 兼容硬件

2.3 适用于开发的库链接

- 用于动态链接的 LightFX 库和标头文件按照 SDK 5.2 内容提供。
- 如果游戏开发人员需要使用 LIB 文件进行静态链接，请联系 Alienware 伙伴关系经理分发文件并获取使用说明。

2.3.1 显式动态链接

使用显式动态链接的好处是开发者即使在系统上没有可用的库时（例如 Alienware Command Center 不存在时）也能够启动应用程序。函数名称和函数指针的定义已经包含在显式动态链接的库标头文件当中。这种库通过调用 LoadLibrary 以及 GetProcAddress 进行加载（而不是使用导入库），以便获取库中的任何所需函数。

2.4 Alienware AlienFX SDK 工作原理

Alienware AlienFX 是一种抽象和转换库，用于与平台硬件和/或已连接的 Alienware 设备的照明系统进行通信。Alienware AlienFX 支持多种设备通信协议并提供通用功能子集，用于为已连接到系统的 RGB 灯（LED 或其他类型）获取并设置颜色值。初始化后，LightFX 模块库会确定所有已连接到系统并启用 Alienware AlienFX 的硬件或灯光控制器，并将上述内容以及它们相对于机械机柜（例如机箱）的物理位置（之中、之上、附近或相连）构建一个列表。

确定所有硬件后，Alienware AlienFX 会通过 AlienFX SDK 提供的导出函数等待来自应用程序的请求。在简单的函数（如 LFX_Light）当中，有关状态变化的所有决策均由库做出，而在复杂的函数（如 LFX_SetLightColor）当中，则需要为有效设备和有效灯光请求索引，以便设置颜色值。

2.5 在 Alienware AlienFX SDK 中进行“Hello World”编程

尽管使用颜色值代替字符串，所需的编程示例“hello world”也能够在 Alienware AlienFX 中实施。以下是其详细内容：

```
LFX_Initialize();

// 将状态机中的所有灯设置为蓝色
LFX_Light(LFX_ALL, LFX_BLUE | LFX_FULL_BRIGHTNESS);

// 导致物理颜色改变
LFX_Update();

// 使系统等待以获取视觉反馈
Sleep(100)

// 清理并离开系统
LFX_Release();
```

虽然本示例只是设置颜色并立即退出（因此会还原以前的状态），但它还是能展现出将 Alienware AlienFX 支持整合到应用程序当中是一件非常简单的事情。灯光更新循环会按照固定应用程序间隔执行。此外，调用 LFX_Light 受限于通过调用 LFX_Update 进行排队并提交至硬件的事件。请参阅本文的函数参考部分，了解有关这些函数及其参数的更多详情。

3 应用程序开发指南

下列指南可供参考。

3.1 弃用函数

- **LFX_UpdateDefault：**此函数已弃用。仅可通过 AWCC 的 FX 模块完成此操作
- **LFX_SetTiming：**此函数已弃用。仅可通过 AWCC 的 FX 模块完成此操作
- **AlienFX Configurator** 已弃用并从 SDK 中移除
- **SDK 受管库** 已弃用并从 SDK 中移除

3.2 位置及定位语义

描述物理位置时，Alienware AlienFX 使用逻辑等值，如 front-Lower-Left。可在标头文件中找到逻辑位置含义的详细信息。

3.3 多线程和命令定时

Alienware AlienFX 将重要的部分整合到每个导出的库函数当中。

此外，Alienware AlienFX 硬件抽象层整合了命令队列和命令处理程序线程，这会掩盖硬件延迟。

这些硬件延迟因设备而异，因此命令处理程序会监测硬件的性能并放弃耗时过长的更新，从而使请求物理变化的软件窗口保持紧凑状态。当前，此窗口设置为 100 毫秒，这意味着如果更新位于命令队列超过 100 毫秒，系统将放弃此更新，以允许硬件与软件保持一致。这种命令缓冲方法可确保核心 Alienware AlienFX 函数（LFX_Light、LFX_SetLightColor、LFX_Update）不会阻止主应用程序线程。

注意：如果命令排队超过 100 毫秒，它们将不会应用到硬件。

3.4 即插即用功能

Alienware AlienFX 将在确定新设备之后实时启用。如果应用程序希望在“运行过程中”添加新 Alienware AlienFX 设备，则可通过调用 LFX_GetNumDevices(..) 以获取新的枚举设备来实现此目标。请注意，如果您在通过不同的函数设置颜色时引用“所有”设备，则无需重新枚举（调用 LFX_GetNumDevices(..)）

4 函数参考

4.1 概览

所有未受管的 Alienware AlienFX 库函数将导出为 C 语言，并且全部返回 LFX_RESULT（类型定义为无符号整数）。以下各节概述了合规 Alienware AlienFX 库中可用的最小强制函数集。

4.2 LFX_Initialize

此函数会初始化 Alienware AlienFX 系统。必须在调用其他库函数之前调用此函数。如果未调用此函数，则系统不会初始化，并且其他库函数将返回 LFX_ERROR_NOINIT or LFX_FAILURE。

语法：

LFX_RESULT LFX_Initialize();

参数：

None

输入：

None

输出：

None

返回：

LFX_SUCCESS	如果系统成功初始化或已初始化
LFX_FAILURE	如果初始化失败
LFX_ERROR_NODEVS	如果系统成功初始化，但没有可用设备

4.3 LFX_Release

此函数会释放 Alienware AlienFX 系统，从而释放内存并将系统还原至其初始状态。当不再需要系统时可能会调用此函数。

即插即用注意事项：应用程序可能会选择释放系统并重新初始化，以响应设备确定通知。这样做会导致在应用程序运行过程中添加新设备。

语法：

LFX_RESULT LFX_Release();

参数：

None

输入：

None

输出：

None

返回：

LFX_SUCCESS	如果系统成功释放

4.4 LFX_Reset

此函数将 Alienware AlienFX 系统中的所有灯设置为“关闭”或无色状态。必须注意：物理灯不会立即发生改变。只有调用 LFX_Update 函数后才会发生改变。例如，要禁用所有灯，请调用 LFX_Reset，然后调用 LFX_Update。

语法：

LFX_RESULT LFX_Reset();

参数：

None

输入：

None

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果没有可用于重设的设备
LFX_SUCCESS	如果重设成功

4.5 LFX_Update

此函数通过向硬件提交任何状态改变来更新 Alienware AlienFX 系统。

语法：

LFX_RESULT LFX_Update();

参数：

None

输入：

None

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果没有可用于重设的设备
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果更新成功

4.6 LFX_UpdateDefault

此函数通过向硬件提交任何状态改变来更新 Alienware AlienFX 系统，并且设置相应的标志，从而将已更新的状态启用为新的开机默认状态。

注意：此函数已弃用

语法：

LFX_RESULT LFX_UpdateDefault();

参数：

None

输入：

None

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果没有可用于重设的设备
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.7 LFX_GetNumDevices

此函数提供连接到系统的 Alienware AlienFX 设备数量。

语法：

LFX_RESULT LFX_GetNumDevices (unsigned int* const numDevices);

参数：

numDevices	即将使用设备数量填充的整数

输入：

None

输出：

使用当前已连接的设备数量填充无符号整数

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果没有可用于重设的设备
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.8 LFX_GetDeviceDescription

此函数获取一个连接到系统的设备的说明和类型。

语法：

```
LFX_RESULT LFX_GetDeviceDescription(const unsigned int devIndex,
        char* const devDesc, const unsigned int devDescSize,
        unsigned char* const devType);
```

参数：

devIndex	索引至目标设备
devDesc	即将使用目标设备说明填充的字符阵列
devDescSize	devDesc 中提供的字符阵列大小
devType	即将使用设备类型填充的无符号短整数

输入：

接受设备索引

输出：

使用编有索引的设备说明填充字符阵列

使用设备类型填充一个无符号短整数

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果索引中没有设备
LFX_ERROR_BUFFSIZE	如果提供的字符阵列太小
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.9 LFX_GetNumLights

此函数返回系统中已连接到设备的 Alienware AlienFX 灯数量。

原型：

```
LFX_RESULT LFX_GetNumLights(const unsigned int devIndex,
        unsigned int* const numLights);
```

参数：

devIndex	索引至设备
numLights	即将使用设备索引处灯数量填充的无符号整数

输入：

接受设备索引

输出：

使用当前在给定索引处连接至设备的灯数量填充无符号整数

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果索引中没有设备
LFX_ERROR_NOLIGHTS	如果在提供的设备索引处没有可用的灯
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.10 LFX_GetLightDescription

此函数获取一个连接到系统的灯的说明。

语法：

```
LFX_RESULT LFX_GetLightDescription(const unsigned int devIndex,
        const unsigned int lightIndex, char* const lightDesc,
        const unsigned int lightDescSize);
```

参数：

devIndex	索引至目标设备
lightIndex	索引至目标灯
lightDesc	即将使用目标灯说明填充的字符阵列
lightDescSize	lightDesc 中提供的字符阵列大小

输入：

接受设备索引

接受灯索引

输出：

使用编有索引的灯说明填充字符阵列

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果索引中没有设备
LFX_ERROR_NOLIGHTS	如果在提供的设备索引处没有可用的灯
LFX_ERROR_BUFFSIZE	如果提供的字符阵列太小
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.11 LFX_GetLightLocation

此函数获取一个连接到系统的灯的位置。

语法：

```
LFX_RESULT LFX_GetLightLocation(const unsigned int devIndex,
        const unsigned int lightIndex, PLFX_POSITION const lightLoc);
```

参数：

devIndex	索引至目标设备
lightIndex	索引至目标灯
lightLoc	即将使用灯位置填充的 LFX_POSITION 结构指针

输入：

接受设备索引

接受灯索引

输出：

使用编有索引的灯位置填充 LFX_POSITION 结构

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果索引中没有设备
LFX_ERROR_NOLIGHTS	如果在提供的设备索引处没有可用的灯
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.12 LFX_GetLightColor

此函数获取一个连接到系统的灯的颜色。此函数提供活动状态中存储的当前颜色。它不一定表示物理灯的颜色。要确保返回的值表示物理灯的状态，请在调用 LFX_Update 后立即调用此函数。

语法：

```
LFX_RESULT LFX_GetLightColor(const unsigned int devIndex,
    const unsigned int lightIndex, PLFX_COLOR const lightCol);
```

参数：

devIndex	索引至目标设备
lightIndex	索引至目标灯
lightCol	即将使用灯位置填充的 LFX_COLOR 结构指针

输入：

接受设备索引

接受灯索引

输出：

使用编有索引的灯颜色填充 LFX_COLOR 结构

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果索引中没有设备
LFX_ERROR_NOLIGHTS	如果在提供的设备索引处没有可用的灯
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.13 LFX_SetLightColor

此函数向命令队列提交灯命令，这会将灯的当前颜色设置为提供的颜色值。此函数会更改自上次重设以来存储在活动状态中的当前颜色。它不会立即更新物理灯设置，而是需要调用 LFX_Update。

语法：

```
LFX_RESULT LFX_SetLightColor(const unsigned int devIndex,
        const unsigned int lightIndex, const PLFX_COLOR lightCol);
```

参数：

devIndex	索引至目标设备
lightIndex	索引至目标灯
lightCol	即将使用灯位置填充的 LFX_COLOR 结构指针

输入：

接受设备索引
接受灯索引
接受 LFX_COLOR 结构指针

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NODEVS	如果索引中没有设备
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.14 LFX_Light

此函数向命令队列提交灯命令，这会将指定位置掩码内任何灯的当前颜色设置为提供的颜色设置。与 LFX_SetLightColor 相似，这些设置在活动状态中更改，并且必须在调用 LFX_Update 提交。位置掩码是一个 32 位的字段，其中前 27 位的每一位都代表虚拟立方体（表示系统）中的一个区域。颜色打包为一个 ARGB 32 位值，其中 alpha 值与亮度相对应。

语法：

```
LFX_RESULT LFX_Light(const unsigned int locationMask,
        const unsigned int colorVal);
```

参数：

locationMask	32 位位置掩码。请参阅标头文件 (LFXDecl.h) 中的定义值
colorVal	32 位颜色值

输入：

接受 32 位位置掩码。
接受 32 位打包颜色值

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NOLIGHTS	如果指定位置掩码处没有发现灯
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.15 LFX_SetLightActionColor

此函数设置灯的主要颜色和动作类型。它会更改自上次 LFX_Reset() 调用后存储在活动状态中的当前颜色和动作类型。它不会立即更新物理灯设置，而是需要调用 LFX_Update()。如果动作类型是一个变体，则动作的次要颜色为黑色。

语法：

```
LFX_RESULT LFX_SetLightActionColor(const unsigned int devIndex,
    const unsigned int lightIndex, const unsigned int actionType,
    const PLFX_COLOR primaryColor);
```

参数：

devIndex	索引至目标设备
lightIndex	索引至目标灯
actionType	动作类型
primaryColor	LFX_COLOR 结构指针（带有所需颜色）

输入：

接受设备索引、灯索引、动作类型 (LFX_ACTION_MORPH, LFX_ACTION_PULSE, LFX_ACTION_COLOR) 和一个新的主 LFX_COLOR 值

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.16 LFX_SetLightActionColorEx

此函数设置灯的主要和次要颜色以及动作类型。它会更改自上次 LFX_Reset() 调用后存储在活动状态中的当前颜色和动作类型。它不会立即更新物理灯设置，而是需要调用 LFX_Update()。如果动作类型不是一个变体，则忽略次要颜色。

语法：

```
LFX_RESULT LFX_SetLightActionColorEx(const unsigned int devIndex,
    const unsigned int lightIndex, const unsigned int actionType,
    const PLFX_COLOR primaryColor, const PLFX_COLOR secondaryColor);
```

参数：

devIndex	索引至目标设备
lightIndex	索引至目标灯
actionType	动作类型
primaryColor	LFX_COLOR 结构指针（带有所需颜色）
secondaryColor	LFX_COLOR 结构指针（带有所需次要颜色）

输入：

接受设备索引、灯索引、动作类型 (LFX_ACTION_MORPH, LFX_ACTION_PULSE, LFX_ACTION_COLOR) 和两个 LFX_COLOR 值

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.17 LFX_ActionColor

此函数为某个位置上带有灯的任何设备设置主要颜色和动作类型。它会更改自上次 LFX_Reset() 调用后存储在活动状态中的当前主要颜色和动作类型。它不会立即更新物理灯设置，而是需要调用 LFX_Update()。如果动作类型是一个变体，则动作的次要颜色为黑色。位置掩码是一个 32 位的字段，其中前 27 位的每一位都代表虚拟立方体（表示系统）中的一个区域。颜色打包为一个 ARGB 32 位值，其中 alpha 值与亮度相对应。

语法：

```
LFX_RESULT LFX_ActionColor(const unsigned int locationMask,
        const unsigned actionType, const unsigned int primaryColor);
```

参数：

locationMask	32 位位置掩码。请参阅标头文件 (LFXDecl.h) 中的定义值
actionType	动作类型
primaryColor	32 位颜色值

输入：

接受 32 位位置掩码

接受 32 位打包颜色值

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NOLIGHTS	如果指定位置掩码处没有发现灯。
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.18 LFX_ActionColorEx

此函数为某个位置上带有灯的任何设备设置主要颜色和次要颜色以及动作类型。它会更改自上次 LFX_Reset() 调用后存储在活动状态中的当前主要颜色和次要颜色以及动作类型。它不会立即更新物理灯设置，而是需要调用 LFX_Update。如果动作类型不是一个变体，则忽略次要颜色。位置掩码是一个 32 位的字段，其中前 27 位的每一位都代表虚拟立方体（表示系统）中的一个区域。颜色打包为一个 ARGB 32 位值，其中 alpha 值与亮度相对应。

语法：

```
LFX_RESULT LFX_ActionColorEx(const unsigned int locationMask,
        const unsigned actionType, const unsigned int primaryColor,
        const unsigned int secondaryColor);
```

参数：

locationMask	32 位位置掩码。请参阅标头文件 (LFXDecl.h) 中的定义值
actionType	动作类型
primaryColor	32 位主要颜色值
secondaryColor	32 位次要颜色值

输入：

接受 32 位位置掩码
接受 32 位打包颜色值

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_NOLIGHTS	如果指定位置掩码处没有发现灯。
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.19 LFX_SetTiming

此函数更改用于下个动作的当前速度或定时。它不会立即更新物理灯设置，而是需要调用 LFX_Update()。定时是指每台设备允许的最低速度和最高速度之间的值。如果输入的值小于最小值或大于最大值，则会将值重新调整为上述极值。

注意：此函数已弃用

语法：

LFX_RESULT LFX_SetTiming(const int timing);

参数：

定时	32 位定时值（以毫秒为单位）

输入：

接受 32 位定时值

输出：

None

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

4.20 LFX_GetVersion

此函数获取安装在系统中的 SDK 版本。

语法：

```
LFX_RESULT LFX_GetVersion(char* const version,
    const unsigned int versionSize);
```

参数：

version	即将使用版本填充的字符阵列
versionSize	版本中提供的字符阵列大小

输入：

接受故障程序和缓冲区大小

输出：

使用 SDK 版本填充字符阵列

返回：

LFX_ERROR_NOINIT	如果系统未初始化
LFX_ERROR_BUFFSIZE	如果提供的字符阵列太小
LFX_FAILURE	如果发生其他错误
LFX_SUCCESS	如果函数运行成功

注意：如果未找到 LFX_GetVersion 函数，则 SDK 版本为 1.0 或 2.x

A 常见问题解答

问：我在哪里可以找到用于测试的 SDK 说明文件、示例、标头文件和/或 DLL？
答：SDK 通过 AWCC 安装部署在系统中。请查看 SDK 5.2 内容 中的位置。

问：应用程序如何检查是否安装了 AlienFX SDK？
答：确认 Windows System 文件夹中存在 LightFX.dll。

问：应用程序安装程序是否应该分发 SDK 的 DLL 和/或任何与 SDK 相关的项目？
答：不应该。应用程序安装程序不应分发 SDK 的 DLL 或任何与 SDK 相关的项目。分发 DLL 会使应用程序灯光实施无法正常运行。

问：为何函数 LFX_xxxx 无法正常工作？
答：虽然 AlienFX SDK 在函数层面依然保持向后兼容，但是如果函数的使用率非常低或者函数在架构层面变得过时，则部分函数就会被弃用。请参阅弃用函数部分，了解弃用的函数和项目列表

问：应用程序发送更新命令的频率如何？
答：为获得最佳性能，请按照多线程部分中的原则进行操作。

问：应该单独控制 LED 还是将其作为一个整体进行控制？
答：在尝试设置相同的系统颜色时，请使用推荐的函数为所有 LED 寻址，而不是单独为每个 LED 寻址。将灯光系统视为一个整体总会事半功倍。为 LED 寻址的推荐函数是位置参数中的 LFX_Light 以及 LFX_All 掩码。此组合性能最佳。当然，SDK 允许您将命令发送到单独的 LED，但是开发人员需要考虑带有 4 个 LED 的系统与带有 100 多个 LED 的系统哪个性能效果更加出色。此外，性能还会受到更新循环速度的影响。每 100 毫秒发送一次命令的效率要远远高于每 20 毫秒发送一次，当更改时间快于用户目测时尤为如此。

问：能否在搭载 Alienware Command Center 4.x 或更早版本的旧 Alienware 系统中进行开发？
答：不能。仅支持使用 Alienware Command Center 5.2 或更高版本进行开发。

问：使用 SDK 5.2 完成的开发是否会影响使用旧 SDK 的最终用户？
答：不会。AlienFX SDK 5.2 与旧 SDK 版本向后兼容。

问：如何在 Unity 中使用 SDK？
答：有 Unity 社区维护的 AlienFX SDK 包装，请联系 Alienware 合作伙伴经理了解详情。

问：是否有适用于《虚幻引擎》的 SDK 包装？
答：是的，有适用于《虚幻引擎》的 AlienFX SDK 包装，请联系 Alienware 合作伙伴经理了解详情。

B 故障排除

- **应用程序在 SDK v1.x 和 v2.x 上工作正常，但无法在 v5.2 上工作**
  检查应用程序是否将 LightFX.dll 作为其文件的一部分进行部署。

- **在用户系统中安装后，应用程序无法控制灯**
  检查用户是否拥有兼容硬件以及针对该硬件安装了最新的 Alienware Command Center

- **机箱灯光工作正常，但外围设备灯光无法正常工作**
  - 检查系统是否为外围设备安装了最新的驱动程序。前往 www.dell.com 下载最新版本
  - 检查 Alienware Command Center 是否能够控制灯光系统；如果不能，请致电 Alienware 支持服务。

- **由应用程序控制的灯光正在发送命令，但硬件无法反映出这些更改或仅能反映部分更改**
  - 是否调用了 LFX_Update 函数？请参阅函数参考中的 LFX_Update。
  - 应用程序发送命令是否过快并且/或为每个 LED 单独寻址？
    - SDK 不保证每个命令都将影响的硬件，因为部分命令会被丢弃。如果是这种情况，应用程序开发人员应该通过增加命令之间的延迟或使用 LFX_Light 函数一次性为所有 LED 寻址来减少发送的命令总数。请参阅即插即用以及 LFX_Light 函数部分了解详情。
    - 此外，开发人员应谨记，在发送命令时，并不是所有硬件都是相同的。在带有 4 个 LED 的系统中为每个 LED 单独寻址与在带有 100 多个 LED 的系统中寻址是完全不同的事情

- **应用程序在开发过程中崩溃**
  检查是否遵循系统要求部分中的规格以及使用的 LightFX.dll 是否由 Alienware Command Center 5.2 部署