Alienware AlienFX SDK 5.2

ユーザーズガイド

概要

これはソフトウェア開発キット（SDK）です。ここにはアプリケーション開発者が、AlienFX ハードウェア プラットフォームおよび AlienFX 互換デバイスが接続されたプラットフォームでのカラー ライトの設定に使用できる機能や関数が用意されており、さまざまな視覚効果を実装できます。

2019 年 3 月

リビジョン

バージョン	SDK バージョン	リリース日	注記
2.0	5.2	2019 年 3 月 6 日	• DLL および関数について、廃止されたものと管理中のものを明確化 • 最新バージョン SDK のサポートについての開発要件を変更 • FAQ およびトラブルシューティングに関する付録を追加
1.2	2.1	2012 年 3 月 26 日	LFX_GetVersion 関数を追加
1.1	2.0	2011 年 5 月 1 日	ゲーム コンフィギュレーター機能を追加
1.0	1.0	2009 年 3 月 24 日	アルファ リリース

目次

1 前書

1.1 対象ユーザー

Dell Alienware AlienFX ソフトウェア開発キット（SDK）の想定するユーザーは、AlienFX ハードウェア プラットフォームおよび AlienFX 互換デバイスの接続されたプラットフォーム用アプリケーションの開発者です。

メモ：AlienFX SDK 5.2 で開発されるアプリには、以前の AlienFX SDK を使用したシステムとの下位互換性があります。

1.2 目的

本文書の目的は、Alienware AlienFX SDK で利用可能な機能と関数について、詳細なプログラム上のアウトラインを提供することです。これらを利用することでアプリケーション開発者は、システムに接続された Alienware AlienFX デバイスのライトを制御できます。Alienware AlienFX エコシステムを制御することで、個々のアプリケーションで要求されるさまざまな視覚効果を、デバイスのカラー ライトとして構成できます。

1.3 スコープ

本文書では、LightFX.dll ライブラリーで利用可能な機能として、各種の関数、依存ライブラリーおよび、システムでのカラー ライト操作に必要なデータとメソッドなどを解説しています。

メモ：AWCC 4.x で導入される AlienFX SDK 2.1 以前での開発はサポートされていません。

2 はじめに

2.1 SDK 5.2 の内容

このソフトウェア開発キット（SDK）は、ライト ライブラリー、アプリケーション サンプル、本ドキュメントで構成されています。Alienware Command Center をインストールすると、インストール ディレクトリーのサブディレクトリー（C:\Program Files\Alienware\Command Center）に、表 1 のファイルが取り込まれます。

Table 1 アーティファクト ライブラリーとロケーション

アーティファクト	ロケーション
マニュアル	[InstallDir]\AlienFX SDK\
ヘッダーと同梱ファイル	[InstallDir]\AlienFX SDK\includes\
動的リンク ライブラリー	• [InstallDir]\AlienFX SDK\DLLs\x86\LightFX.dll • [InstallDir]\AlienFX SDK\DLLs\x64\LightFX.dll 同じライブラリーは下記にもあります。 • \Windows\System32\LightFX.dll （64 ビット バージョン） • \Windows\SysWOW64\LightFX.dll （32 ビット バージョン）
サンプル	[InstallDir]\AlienFX SDK\Samples\

警告： Alienware AlienFX SDK で開発したアプリケーションには、上記のライブラリーを同梱しないようにしてください。最終リリース版のアプリケーションでは、Alienware Command Center で導入される LightFX.dll を使用する必要があります。

2.2 開発用のシステム要件

AlienFX プラットフォームおよび周辺機器の 2018 年以降のリリースでは、AWCC 5.2 以降をインストールして AlienFX SDK 5.2 での開発を行う必要があります。

Table 2 開発用のシステム要件

タイプ	要件
オペレーティングシステム	AlienFX SDK 5.x：Windows 10、64 ビット版
ソフトウェア	Alienware Command Center 5.2.x
ハードウェア	Alienware AlienFX 5.2 準拠ハードウェア

2.3 開発用ライブラリーのリンク

- 提供される LightFX ライブラリーおよび動的リンク用のヘッダー ファイルは、「SDK 5.2 の内容」に記載されているとおりです。
- ゲーム開発者として静的リンク用の LIB ファイルの使用を必要とする場合は、Alienware のパートナーシップ マネージャーにファイルの配布と使用方法についての説明をお問い合わせください。

2.3.1 明示的な動的リンク

明示的な動的リンクを使用するメリットは、Alienware Command Center のないシステムなど、ライブラリーが使用できない場合でもアプリケーションを起動できることです。明示的な動的リンクのライブラリーでは、そのヘッダー ファイル中に関数の名前および関数ポインターの定義が含まれています。インポート ライブラリーを使用する代わりに、ライブラリーのロード時には LoadLibrary の呼び出しが行われ、必要とされる任意のライブラリー関数の GetProcAddress が呼び出されます。

2.4 Alienware AlienFX SDK の仕組み

Alienware AlienFX は 1 つの抽象化および翻訳ライブラリーであり、プラットフォーム ハードウェアおよび接続された Alienware デバイスのライト システムとの通信に使用されます。Alienware AlienFX では、さまざまなデバイス通信プロトコルがサポートされており、システムに接続された RGB ライト（LED その他のタイプ）のカラー値の取得と設定を行うための共通機能のサブセットが提供されています。初期化後の LightFX モジュール ライブラリーは、システムに接続されたすべての Alienware AlienFX 対応ハードウェアまたはライト コントローラーを識別し、それらのリストを、機械的エンクロージャー（シャーシ）に対する物理的な接続位置と併せて作成します。

すべてのハードウェアの識別後に Alienware AlienFX は、AlienFX SDK の提供する関数エクスポートを通じて、アプリケーションからの要求を待機します。こうした関数には、LFX_Light のように、すべての状態変更についての判定がライブラリーで処理される単純なものもあれば、LFX_SetLightColor のように、カラー値を設定する有効なライトおよび有効なデバイスのインデックスという詳細な記述を必要とするものもあります。

2.5 Alienware AlienFX SDK を使用した「Hello World」

プログラマーにお馴染みのサンプルである「Hello World」は Alienware AlienFX でも実装できますが、ここでは文字列ではなくカラー値で表現します。具体的には、次のように記述します。

```
LFX_Initialize();

//ステート マシンですべてのライトを青色に設定します
LFX_Light(LFX_ALL, LFX_BLUE | LFX_FULL_BRIGHTNESS);

//物理的に色を変化させます
LFX_Update();

//視覚的なフィードバックが確立するまでシステムを待機させます
Sleep(100)

//クリーンアップしてシステムから離れます
LFX_Release();
```

このサンプルは、ごく単純に色を設定してすぐに終了する（実行前の状態に復元する）だけですが、アプリケーションへの Alienware AlienFX サポートの組み込みが非常に簡単なことを示すデモでもあります。ライトおよび更新のループは、通常のアプリケーション インターバルと並行して実行できます。また LFX_Light の呼び出しをキューのイベントに関連付けることもでき、ハードウェアへの送信は LFX_Update の呼び出し時に行われます。これらの関数およびパラメーターの詳細については、本文書の「関数リファレンス」のセクションを参照してください。

3 アプリケーション開発ガイドライン

以下のガイドラインは、参照用に用意したものです。

3.1 廃止された関数

- **LFX_UpdateDefault：**この関数は廃止されました。この処理は AWCC の FX モジュールを介してのみ行えます。
- **LFX_SetTiming：**この関数は廃止されました。この処理は AWCC の FX モジュールを介してのみ行えます。
- **AlienFX Configurator** は廃止され、SDK から削除されました
- **SDK 管理ライブラリー**は廃止され、SDK から削除されました

3.2 ロケーションとポジショニングのセマンティック

Alienware AlienFX では、物理的なロケーションの記述に、front-Lower-Left などの論理的に等価な表記を使用します。論理ロケーションの意味の詳細については、ヘッダー ファイルを参照してください。

3.3 マルチスレッドとコマンドのタイミング

Alienware AlienFX では、エクスポートされる個々のライブラリー関数に重要なセクションが組み込まれています。

また Alienware AlienFX のハードウェア抽象化レイヤーでは、ハードウェア レイテンシーをマスクするコマンド ハンドラー スレッドおよびコマンド キューが組み込まれています。

こうしたハードウェア レイテンシーはデバイスごとに異なるため、コマンド ハンドラーは、ハードウェアのパフォーマンスを監視し、時間がかかりすぎる更新をドロップして、物理的な変更に対するソフトウェア要求のウィンドウが厳密に維持されるようにします。現状このウィンドウ設定は 100 ミリ秒であり、コマンド キューにある更新が 100 ミリ秒を超過した場合はドロップして、ハードウェアがソフトウェアに追いつけるようにします。このようなコマンド バッファリングの手法により、Alienware AlienFX のコア関数（LFX_Light、LFX_SetLightColor、LFX_Update）がメイン アプリケーション スレッドをブロックする事態を防止しています。

メモ：100 ミリ秒を超えてキューに置かれたコマンドは、ハードウェアに適用されません。

3.4 プラグ アンド プレイ機能

Alienware AlienFX は新しいデバイスの接続をリアル タイムに監視します。新しい Alienware AlienFX デバイスを「その場」で追加したいアプリケーションの場合、そうした処理は LFX_GetNumDevices(..)を呼び出して新しい列挙型デバイスを取得することで行えます。異なる関数を使用して色を設定する場合における、「すべて」のデバイスの参照では、再列挙（LFX_GetNumDevices(..)の呼び出し）を行う必要はありません。

4 関数リファレンス

4.1 概要

すべての管理されていない Alienware AlienFX ライブラリー関数は C 言語としてエクスポートされ、いずれも LFX_RESULT（定義型は符号なし整数）を返します。以下のセクションでは、Alienware AlienFX の準拠ライブラリーで使用可能な必要最低限の関数セットについて説明します。

4.2 LFX_Initialize

この関数は Alienware AlienFX システムを初期化します。この呼び出しは、他のライブラリー関数を呼び出す前に行う必要があります。この関数が呼び出されていない場合、システムは初期化されず、他のライブラリー関数からは LFX_ERROR_NOINIT or LFX_FAILURE が返されます。

構文：

LFX_RESULT LFX_Initialize();

パラメーター：

None

入力：

None

出力：

None

以下を返します：

LFX_SUCCESS	システムが正常に初期化された場合、またはすでに初期化されている場合
LFX_FAILURE	初期化に失敗した場合
LFX_ERROR_NODEVS	システムは初期化されているが、使用可能なデバイスがない場合

4.3 LFX_Release

この関数は、Alienware AlienFX システムをリリースしてメモリーを解放し、システムを初期状態に戻します。呼び出すのは、システムが不要になった時点です。

プラグ アンド プレイについてのメモ：デバイスの接続通知に応じて、アプリケーションがシステムのリリースと再初期化を選択するようにすることもできます。これにより、アプリケーションの実行中に追加された新しいデバイスが考慮されるようになります。

構文：

LFX_RESULT LFX_Release();

パラメーター：

None

入力：

None

出力：

None

以下を返します：

LFX_SUCCESS	システムが正常にリリースされた場合

4.4 LFX_Reset

この関数は、Alienware AlienFXシステム内のすべてのライトを「オフ」または無色の状態に設定します。物理的な光源に対する変更は、すぐには反映されない点には注意が必要です。変更が生じるのは、LFX_Update関数を呼び出した後のみです。たとえば、すべてのライトを無効にするには、LFX_Resetに続けてLFX_Updateを呼び出します。

構文：

LFX_RESULT LFX_Reset();

パラメーター：

None

入力：

None

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	リセットできるデバイスがない場合
LFX_SUCCESS	リセットが正常に完了した場合

4.5 LFX_Update

この関数は、ハードウェアの状態変更を送信することでAlienware AlienFXシステムを更新します。

構文：

LFX_RESULT LFX_Update();

パラメーター：

None

入力：

None

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	リセットできるデバイスがない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	更新が正常に完了した場合

4.6 LFX_UpdateDefault

この関数は、ハードウェアの状態変更を送信することによって Alienware AlienFX システムを更新し、また適切なフラグを設定することにより、更新された状態を新しいパワーオン デフォルト状態にします。

メモ：この関数は廃止されました。

構文：

LFX_RESULT LFX_UpdateDefault();

パラメーター：

None

入力：

None

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	リセットできるデバイスがない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.7 LFX_GetNumDevices

この関数は、Alienware AlienFX システムに接続されているデバイスの数を提供します。

構文：

LFX_RESULT LFX_GetNumDevices (unsigned int* const numDevices);

パラメーター：

numDevices	デバイス数が入力される整数

入力：

None

出力：

現在接続されているデバイス数が、符号なし整数に入力されます

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	リセットできるデバイスがない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.8 LFX_GetDeviceDescription

この関数は、システムに接続されているデバイスの説明およびタイプを取得します。

構文：

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

パラメーター：

devIndex	ターゲット デバイスのインデックス
devDesc	ターゲット デバイスの説明が入力される文字配列
devDescSize	devDescの文字配列のサイズ
devType	デバイス タイプが入力される符号なし短整数

入力：

デバイスのインデックスを受け取ります

出力：

インデックスされたデバイスの説明が文字配列に入力されます

デバイス タイプが符号なし短整数に入力されます

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	インデックスにデバイスがない場合
LFX_ERROR_BUFFSIZE	提供された文字配列が小さすぎる場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.9 LFX_GetNumLights

この関数は、システム内のデバイスに接続されている Alienware AlienFX ライトの数を返します。

プロトタイプ：

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

パラメーター：

devIndex	デバイスのインデックス
numLights	デバイス インデックスにあるライト数が入力される符号なし整数

入力：

デバイスのインデックスを受け取ります

出力：

指定されたインデックスにおいてデバイスに接続されている現在のライト数が、符号なし整数に入力されます

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	インデックスにデバイスがない場合
LFX_ERROR_NOLIGHTS	提供されたデバイス インデックスの使用可能なライトがない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.10 LFX_GetLightDescription

この関数は、システムに接続されているライトの説明を取得します。

構文：

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

パラメーター：

devIndex	ターゲット デバイスのインデックス
lightIndex	ターゲットとなるライトのインデックス
lightDesc	ターゲットとなるライトの説明が入力される文字配列
lightDescSize	lightDescの文字配列のサイズ

入力：

デバイスのインデックスを受け取ります
ライトへのインデックスを受け取ります

出力：

インデックスされたライトの説明が文字配列に入力されます

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	インデックスにデバイスがない場合
LFX_ERROR_NOLIGHTS	提供されたデバイス インデックスの使用可能なライトがない場合
LFX_ERROR_BUFFSIZE	提供された文字配列が小さすぎる場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.11 LFX_GetLightLocation

この関数は、システムに接続されているライトのロケーションを取得します。

構文：

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

パラメーター：

devIndex	ターゲット デバイスのインデックス
lightIndex	ターゲットとなるライトのインデックス
lightLoc	ライトのロケーションが入力されるLFX_POSITION構造体へのポインター

入力：

デバイスのインデックスを受け取ります
ライトへのインデックスを受け取ります

出力：

インデックスされたライトのロケーションがLFX_POSITION構造体に入力されます

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	インデックスにデバイスがない場合
LFX_ERROR_NOLIGHTS	提供されたデバイス インデックスの使用可能なライトがない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.12 LFX_GetLightColor

この関数は、システムに接続されているライトの色を取得します。この関数を用いることで、アクティブな状態として保持されている現在の色が得られます。ただしこの情報は、必ずしも物理的な光源の色を表しているとは限りません。戻り値が物理的なライトの状態を確実に示すようにするには、この関数呼び出しを LFX_Update の呼び出し直後に行う必要があります。

構文：

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

パラメーター：

devIndex	ターゲット デバイスのインデックス
lightIndex	ターゲットとなるライトのインデックス
lightCol	ライトのロケーションが入力されるLFX_COLOR構造体へのポインター

入力：

デバイスのインデックスを受け取ります
ライトへのインデックスを受け取ります

出力：

インデックスされたライトの色がLFX_COLOR構造体に入力されます

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	インデックスにデバイスがない場合
LFX_ERROR_NOLIGHTS	提供されたデバイス インデックスの使用可能なライトがない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.13 LFX_SetLightColor

この関数は、ライト コマンドをコマンド キューに送信することで、現在のライトの色を、指定された色の値に設定します。前回のリセット以降にアクティブな状態として保持されてきた現在の色は、この関数によって変更されます。これは物理的なライトの設定を即座に更新するものではなく、そのためには LFX_Update の呼び出しが必要です。

構文：

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

パラメーター：

devIndex	ターゲット デバイスのインデックス
lightIndex	ターゲットとなるライトのインデックス
lightCol	ライトのロケーションが入力されるLFX_COLOR構造体へのポインター

入力：

デバイスのインデックスを受け取ります
ライトへのインデックスを受け取ります
LFX_COLOR構造体へのポインターを受け取ります

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NODEVS	インデックスにデバイスがない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.14 LFX_Light

この関数は、ライト コマンドをコマンド キューに送信することで、ロケーション マスク内にある現在のライトの色をすべて、指定された色設定に変更します。LFX_SetLightColor と同様に、これらの設定はアクティブ状態で変更され、LFX_Update への呼び出しと併せて送信する必要があります。ロケーション マスクは 32 ビット フィールドで、これらの先頭 27 ビットは、システムを表す仮想キューブ中の 1 つのゾーンに対応しています。色情報は、1 つの 32 ビット ARGB 値にパックされており、このアルファ値が輝度に対応します。

構文：

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

パラメーター：

locationMask	32ビット ロケーション マスク。ヘッダー ファイル（LFXDecl.h）の定義値を参照してください。
colorVal	32ビット カラー値

入力：

32ビット ロケーション マスクを受け取ります。
32ビットにパックされたカラー値を受け取ります。

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NOLIGHTS	指定されたロケーション マスクにライトが見つからなかった場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.15 LFX_SetLightActionColor

この関数は、プライマリー カラーの設定および、アクション タイプのライトへの設定を行います。前回の LFX_Reset()呼び出し以降にアクティブ状態として格納されている現行の色およびアクション タイプが、これにより変更されます。これは物理的なライトの設定を即座に更新するものではなく、そのためには LFX_Update()の呼び出しが必要です。アクション タイプがモーフィングである場合、アクションのセカンダリー カラーはブラックです。

構文：

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

パラメーター：

devIndex	ターゲット デバイスのインデックス
lightIndex	ターゲットとなるライトのインデックス
actionType	アクション タイプ
primaryColor	目的の色を持つLFX_COLOR構造体へのポインター

入力：

デバイスへのインデックス、ライトへのインデックス、アクション タイプ (LFX_ACTION_MORPH, LFX_ACTION_PULSE, LFX_ACTION_COLOR)、新しいプライマリーLFX_COLOR値を受け取ります

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.16 LFX_SetLightActionColorEx

この関数は、プライマリーとセカンダリー カラーの設定および、アクション タイプのライトへの設定を行います。前回の LFX_Reset()呼び出し以降にアクティブ状態として格納されている現行の色およびアクション タイプが、これにより変更されます。これは物理的なライトの設定を即座に更新するものではなく、そのためには LFX_Update()の呼び出しが必要です。アクション タイプがモーフィングでない場合、セカンダリー カラーは無視されます。

構文：

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

パラメーター：

devIndex	ターゲット デバイスのインデックス
lightIndex	ターゲットとなるライトのインデックス
actionType	アクション タイプ
primaryColor	目的の色を持つLFX_COLOR構造体へのポインター
secondaryColor	セカンダリー カラーを持つLFX_COLOR構造体へのポインター

入力：

デバイスへのインデックス、ライトへのインデックス、アクション　タイプ(LFX_ACTION_MORPH, LFX_ACTION_PULSE, LFX_ACTION_COLOR)、2つのLFX_COLOR値を受け取ります

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.17 LFX_ActionColor

この関数は、特定ロケーションにライトがある任意のデバイスのアクション タイプおよびプライマリー カラーを設定します。前回のLFX_Reset()呼び出し以降にアクティブ状態として格納されている現行のプライマリー カラーおよびアクション タイプが、これにより変更されます。これは物理的なライトの設定を即座に更新するものではなく、そのためには LFX_Update()の呼び出しが必要です。アクション タイプがモーフィングである場合、アクションのセカンダリー カラーはブラックです。ロケーション マスクは 32 ビット フィールドで、これらの先頭 27 ビットは、システムを表す仮想キューブ中の 1 つのゾーンに対応しています。色情報は、1 つの 32 ビット ARGB 値にパックされており、このアルファ値が輝度に対応します。

構文：

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

パラメーター：

locationMask	32ビット ロケーション マスク。ヘッダー ファイル（LFXDecl.h）の定義値を参照してください。
actionType	アクション タイプ
primaryColor	32ビット カラー値

入力：

32ビット ロケーション マスクを受け取ります
32ビットにパックされたカラー値を受け取ります。

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NOLIGHTS	指定されたロケーション マスクにライトが見つからなかった場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.18 LFX_ActionColorEx

この関数は、特定ロケーションにライトがある任意のデバイスのアクション タイプおよびプライマリーとセカンダリー カラーを設定します。前回の LFX_Reset()呼び出し以降にアクティブ状態として格納されている現行のプライマリーとセカンダリー カラーおよびアクション タイプが、これにより変更されます。これは物理的なライトの設定を即座に更新するものではなく、そのためには LFX_Update の呼び出しが必要です。アクション タイプがモーフィングでない場合、セカンダリー カラーは無視されます。ロケーション マスクは 32 ビット フィールドで、これらの先頭 27 ビットは、システムを表す仮想キューブ中の 1 つのゾーンに対応しています。色情報は、1 つの 32 ビット ARGB 値にパックされており、このアルファ値が輝度に対応します。

構文：

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

パラメーター：

locationMask	32ビット ロケーション マスク。ヘッダー ファイル（LFXDecl.h）の定義値を参照してください。
actionType	アクション タイプ
primaryColor	32ビット プライマリー カラー値
secondaryColor	32ビット セカンダリー カラー値

入力：

32ビット ロケーション マスクを受け取ります
32ビットにパックされたカラー値を受け取ります。

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_NOLIGHTS	指定されたロケーション マスクにライトが見つからなかった場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.19 LFX_SetTiming

この関数は、次のアクションで使用する現在のテンポまたはタイミングを変更します。これは物理的なライトの設定を即座に更新するものではなく、そのためには LFX_Update()の呼び出しが必要です。タイミングは、各デバイスで許容される最小と最大テンポの間の値です。入力値が最小値より小さいか最大値より大きい場合、値はこれらの極値に再調整されます。

メモ：この関数は廃止されました。

構文：

```
LFX_RESULT LFX_SetTiming(const int timing);
```

パラメーター：

タイミング	32ビット タイミング値（ミリ秒）

入力：

32ビット タイミング値を受け取ります

出力：

None

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

4.20 LFX_GetVersion

この関数は、システムにインストールされているSDKのバージョンを取得します。

構文：

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

パラメーター：

version	バージョンが入力される文字配列
versionSize	バージョンが入力される文字配列のサイズ

入力：

データ ロガーおよびバッファー サイズを受け取ります

出力：

SDKのバージョンが文字配列に入力されます

以下を返します：

LFX_ERROR_NOINIT	システムが初期化されていない場合
LFX_ERROR_BUFFSIZE	提供された文字配列が小さすぎる場合
LFX_FAILURE	その他のエラーが発生した場合
LFX_SUCCESS	正常に機能した場合

メモ：LFX_GetVersion 関数が見つからない場合の SDK バージョンは 1.0 または 2.x です

A FAQ

Q：SDK のドキュメント、サンプル、ヘッダー ファイル、テスト用 DLL などはどこで入手できますか？

A：SDK のシステムへの導入は、AWCC インストールを通じて実施されます。「SDK 5.2 の内容」の記述を確認してください。

Q：AlienFX SDK がインストールされているかをアプリ側から確認するにはどうすればよいですか？

A：LightFX.dll が Windows のシステム フォルダーにあることを確認してください。

Q：アプリのインストーラーは、SDK の DLL や SDK 関連項目を配布する必要がありますか？

A：いいえ。アプリのインストーラーは SDK の DLL や SDK 関連項目を配布してはいけません。DLL を配布することで、アプリのライト実装は機能しません。

Q：LFX_xxxx という関数が機能しなくなったのはなぜですか？

A：AlienFX SDK は関数レベルで後方互換性を維持していますが、利用率が非常に低くなったり、アーキテクチャー レベルで古くなった関数の一部は、廃止される場合があります。廃止された関数や項目のリストについては、「廃止された関数」のセクションを参照してください。

Q：アプリはどのくらいの頻度で更新コマンドを送信する必要がありますか？

A：最適なパフォーマンスを得るには、マルチスレッドを解説したセクションにあるガイドラインに従ってください。

Q：LED の制御は、個別または全体的に行う必要がありますか？

A：共通のシステム カラーを設定する場合は、各 LED を個別に指定するのではなく、推奨される関数を用いてすべての LED を操作してください。常に好ましい操作法は、ライト システム全体をまとめて扱うことです。LED 群の操作に推奨される関数は LFX_Light であり、これに LFX_All マスクとロケーション パラメーターを指定して使用してください。この組み合わせは最高のパフォーマンスを発揮します。もちろん SDK の機能としては個々の LED ごとにコマンド送信を行うことも可能ですが、開発者としては、LED が 4 個しかないシステムの場合と LED が 100 個以上あるシステムの場合とでのパフォーマンス的な影響も考慮する必要があります。またこうした要件には、更新ループの速度も影響します。コマンド送信は 20 ms 間隔で行うよりも 100 ms 間隔で行う方が効率的であり、これはユーザーが視覚的に検知可能な時間よりも変更時間が短い場合に特に顕著になります。

Q：Alienware Command Center 4.x 以前の古い Alienware システムで開発を行うことはできますか？

A：いいえ。サポートされているのは Alienware Command Center 5.2 以降を使用した開発のみです。

Q：SDK 5.2 を用いた開発は、古い SDK を使用しているエンド ユーザーに影響しますか？

A：いいえ。AlienFX SDK 5.2 には、古い SDK バージョンとの下位互換性があります。

Q：Unity で SDK を使用するにはどうすればよいですか？

A：Unity コミュニティーで管理されている AlienFX SDK ラッパーが存在しています。詳細については Alienware のパートナーシップ マネージャーにお問い合わせください。

Q：Unreal Engine 用の SDK ラッパーはありますか？

A：はい、Unreal Engine 用の AlienFX SDK ラッパーが存在しています。詳細については Alienware のパートナーシップ マネージャーにお問い合わせください。

B トラブルシューティング

- **アプリがSDK v1.xおよびv2.xでは動作するが、v5.2では動作しない**
  アプリがファイルの一部としてLightFX.dllを導入していないことを確認してください。

- **ユーザーのシステムへのインストール後のアプリが、ライトを制御できない**
  ユーザーが互換性のあるHWを使用しており、そのHWに対応した最新のAlienware Command Centerがインストールされていることを確認してください。

- **シャーシ ライトは機能するが、周辺ライトが機能しない**
  - 周辺機器用の最新ドライバーがシステムにインストールされていることを確認してください。最新バージョンをダウンロードするにはwww.dell.comにアクセスします。
  - ライト システムがAlienware Command Centerを介して制御できているかを確認し、制御できていない場合はAlienware Support Serviceに問い合わせてください。

- **アプリで制御するライト用のコマンドは送信されているのに、HW側で変更がまったく反映されない、または部分的にしか変更されない**
  - LFX_Update関数は呼び出されていますか？「LFX_Update」の関数リファレンスを参照してください。
  - アプリからのコマンド送信の間隔が速すぎるか、あるいは個々のLEDごとに個別処理をしていませんか？
    - SDKは各コマンドがハードウェアに影響することを想定しており、一部のコマンドは破棄される可能性があります。これに該当する場合、アプリ開発者は、送信コマンド数の削減を試みる必要があり、それにはコマンド送信間の遅延を大きくするか、あるいはLFX_Light関数を使用して全LEDを一括処理します。詳細については、「プラグアンド プレイ」および「LFX_Light関数」のセクションを参照してください。
    - また開発者が注意すべき事項として、コマンド送信時におけるHWがすべて同じとは限らない点もあります。個々のLED別の操作を行う場合、LEDが4個しかないシステムとLEDが100個以上あるシステムとでは処理が異なります。

- **開発中にアプリがクラッシュする**
  「システム要件」のセクションにある仕様が満たされていることおよび、用いるLightFX.dllの導入がAlienware Command Center 5.2を介して行われていることを確認してください。