

# **RGB Fusion**

## **Software Development Kit Guide**

**Version 1.1.6**

**Document Release Date: March 11, 2019**

## Table of Contents

Version History .....	4
1 Introduction .....	5
2 Prerequisites .....	5
3 Motherboard LED Setting Data Reference .....	6
3.1 LED Setting Data Structure .....	6
3.2 wTime & CtrlVal for LED .....	9
4 Motherboard LED Control API Reference .....	17
4.1 The API Use Flow .....	17
4.2 APIs .....	18
4.2.1 dllexp_GetSdkVersion .....	18
4.2.2 dllexp_InitAPI .....	18
4.2.3 dllexp_GetMaxDivision .....	19
4.2.4 dllexp_GetLedLayout .....	19
4.2.5 dllexp_SetLedData .....	20
4.2.6 dllexp_Apply .....	21
4.2.7 dllexp_BeatInput .....	22
4.2.8 dllexp_IT8295_Reset .....	23
4.2.9 dllexp_Get_IT8295_FwVer .....	23
4.3 RGB LED Strip Calibration .....	24
4.3.1 dllexp_RGBCalibration_Step1 .....	26
4.3.2 dllexp_RGBCalibration_Step2.....	26
4.3.3 dllexp_RGBCalibration_Step3 .....	27
4.3.4 dllexp_RGBCalibration_Done .....	27
4.3.5 dllexp_GetCalibrationValue .....	28
4.4 APIs For Monochrome LED .....	28
4.4.1 Monochrome LED API Flow .....	29
4.4.2 dllexp_MonocLedCtrlSupport .....	29
4.4.3 dllexp_GetRGBPinType .....	30
4.4.4 dllexp_SetMonocLedMode .....	30
4.4.5 dllexp_RGBPinType1 .....	31
5 VGA & Peripherals LED Control API Reference .....	32
5.1 APIs .....	32
5.1.1 dllexp_GvLedInitial.....	32
5.1.2 dllexp_GvLedGetVersion .....	34
5.1.3 dllexp_GvLedSet .....	34
5.1.4 dllexp_GvLedGetVgaModelName.....	35



## Version History

Version	Date	Comments
1.1.6	March 11, 2019	1. Fix some driver security issue
1.1.1	March 12, 2018	1. Fix dllexp_BeatInput fail on H310/B360/H370.
1.1.0	February 6, 2018	1. Gigabyte graphics card & Peripherals supported. 2. Intel H310/B360/H370 Series supported. 3. AMD AX470 Series supported.
1.0.0	September 23, 2017	1. First version of RGB Fusion Software Development.

# 1 Introduction

This guide describes how to use the RGB Fusion Software Development Kit (SDK). The purpose of this guide is to provide information to third-party companies that want to control LED lights on motherboards. This guide includes documentation only for the structures and methods that can be used to control LED lights. Third-party software can control the LED lights on the motherboards by employing the methods exported by the components (dll, exe) of this SDK.

# 2 Prerequisites

These are the prerequisites for using this SDK:

- The related components provided by the SDK must have been installed in the computer.
  
- SDK requires the following items to be installed on your computer
  - ◆ Visual Studio 2012 Redistributable (x86/x64)
  
- The components are generated with Microsoft Visual Studio 2012 MFC Project. Third-party software can be developed using MFC or C#.
  
- Supported operating systems
  - ◆ Windows 7 32-bit and 64-bit
  - ◆ Windows 8.1 32-bit and 64-bit
  - ◆ Windows 10 32-bit and 64-bit
  
- Supported MCU:
  - ◆ IT8295
  
- Support Motherboards:
  - ◆ Intel platform: X299/H310/B360/H370/Z370 Series.
  - ◆ AMD platform: AX370/X399/AX470 Series.

## 3 Motherboard LED Setting Data Reference

This chapter introduces the structure used to configure the motherboard LEDs and explains the meaning and usage of each member in the structure.

### 3.1 LED Setting Data Structure

The structure is used to set the motherboard LED mode, color, timing, and some other mode-dependent settings. Each section corresponds to a structure.

#### Syntax

```
typedef struct _LEDSETTING {
    BYTE    Reserve0;
    BYTE    LedMode;
    BYTE    MaxBrightness;
    BYTE    MinBrightness;
    DWORD   dwColor;
    WORD    wTime0;
    WORD    wTime1;
    WORD    wTime2;
    BYTE    CtrlVal0;
    BYTE    CtrlVal1;
} LEDSETTING, *PLEDSETTING
```

#### Members

##### Reserve0

Reserved for future use.

##### LedMode

LED mode. This member can be one of the following values. Default value is 0.

Value	Meaning
0	Null
1	Pulse
2	Music
3	Color Cycle
4	Static
5	Flash
8	Transition
10	Digital Mode A
11	Digital Mode B
12	Digital Mode C
13	Digital Mode D
14	Digital Mode E
15	Digital Mode F
16	Digital Mode G
17	Digital Mode H
18	Digital Mode I

### **MaxBrightness**

LED maximum brightness. The valid values for this member are 0 through 100. Default value is 100.

### **MinBrightness**

LED minimum brightness. The valid values for this member are 0 through 100. Default value is 0.

### **dwColor**

WW, RR, GG, BB have a data length of one byte respectively. Configurable values are 0x0 ~ 0xFF. WW represents white light. Currently, white light is supported only when using an external LED strip. WW=0 is to turn off the white light while WW=0xFF is to turn on the white light. At present there is no function to turn off the LEDs. To turn off the LEDs, you can set LedMode to Static and set this member to 0.

### **wTime0**

LED Timing 0, in millisecond. The valid values for this member are 0 through 65535. Default value is 0. This member represents time parameters with different functions based on the selected LED Mode. This part will be discussed in Chapter 3.2.

### **wTime1**

LED Timing 1, in millisecond. The valid values for this member are 0 through 65535. Default value is 0. This member represents time parameters with different functions based on the selected LED Mode. This part will be discussed in Chapter 3.2.

### **wTime2**

LED Timing 2, in millisecond. The valid values for this member are 0 through 65535. Default value is 0. This member represents time parameters with different functions based on the selected LED Mode. This part will be discussed in Chapter 3.2.

### **CtrlVal0**

LED Control Value 0. The valid values for this member are 0 through 255. Default value is 0. This member represents configuration parameters with different functions based on the selected LED Mode. This part will be discussed in Chapter 3.2.

### **CtrlVal1**

LED Control Value 1. The valid values for this member are 0 through 255. Default value is 0. This member represents configuration parameters with different functions based on the selected LED Mode. This part will be discussed in Chapter 3.2.

## **Remark**

1. The LedMode value start from 10 is digital LED mode that is for digital LED strip used only. Currently there are two kinds of digital LED strips on the motherboard. One is the LED strip integrated in the audio shield for some high-end motherboards supporting the RGB fusion; the other is the digital LED strip purchased by end-user and is connected to the motherboard through the D\_LED header.
2. Not all above-mentioned digital LED modes are supported by the two kinds of LED strips. The table below lists their supported modes:

<b>D_LED_TYPE1 (Audio Shield)</b>	<b>D_LED_TYPE2 (LED Strip)</b>
Digital Mode A, Digital Mode B, Digital Mode E	Digital Mode A, Digital Mode B, Digital Mode C, Digital Mode D, Digital Mode F, Digital Mode G, Digital Mode H, Digital Mode I

## 3.2 wTime & CtrlVal for LED Mode

This section explains the roles of the wTime and CtrlVal parameters in each LED Mode using tables. If the “Value” member in the table shows "X", it means this parameter can be ignored in this LED mode. For all LED modes, the unit for the time parameters wTime0 ~ wTime2 is millisecond.

- Pulse Mode

Name	Value	Description
LedMode	1	Pulse Mode
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Lighting on time of pulsing
wTime1	0-65535	Lighting off time of pulsing
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

N/A.

- Music Mode

Name	Value	Description
LedMode	2	Music Mode
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	X	
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

Music mode is driven by software. The MCU will enter a state where it can only be passively triggered by software after switching to Music mode. In

this moment, all of the LEDs are turned off. Programmers have to capture the rhythm of music and then set the brightness of the LEDs to change with the music beats through the `dllexp_BeatInput` (see 4.2.7) API.

- Color Cycle Mode

Name	Value	Description
LedMode	3	Color Cycle Mode
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Transition time of color change
wTime1	X	
wTime2	X	
CtrlVal0	1-7	Set the number of colors in a cycle.
CtrlVal1	0-1	0 = Color Cycle only 1 = Enable Color Cycle + Pulse

### Remark

N/A.

- Static Mode

Name	Value	Description
LedMode	4	Static Mode
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	X	
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

N/A.

- Flash Mode

Name	Value	Description
LedMode	5	Flash Mode
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Flash on/off time
wTime1	0-65535	Flash interval (Flash off/on time = Flash interval - flash on/off time)
wTime2	0-65535	Flash cycle time
CtrlVal0	1-255	Flash "N" times
CtrlVal1	X	

### Remark

- If MCU firmware version < 0x00020001, then wTime0 = Flash On Time
- If 0x00020001 >= MCU firmware version < 0x00030000, then wTime0 = Flash Off Time.
- Please note that in Flash Mode, wTime0, wTime1, wTime2 must meet the following conditions, otherwise Flash Mode will be unable to display correctly.
  - ◆  $wTime0 < wTime1 \leq wTime2$
  - ◆ If  $CtrlVal0 > 1$ , then  $(CtrlVal0 * wTime1) \leq wTime2$

- Transition Mode

Name	Value	Description
LedMode	9	Transition Mode
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Transition time
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

The effect of this mode is to gradually convert the color from the current color to the dwColor. The Transition time parameter can be used to set the color transition time, but there will be some time difference and may not be 100% accurate.

- Digital Mode A

Name	Value	Description
LedMode	10	Digital Mode A
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	X	
wTime0	0-65535	Speed
wTime1	X	
wTime2	X	
CtrlVal0	0/1	Wave Direction 0 = right to left 1 = left to right
CtrlVal1	X	

### Remark

N/A.

- Digital Mode B

Name	Value	Description
LedMode	11	Digital Mode B
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Speed
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

N/A.

- Digital Mode C

Name	Value	Description
LedMode	12	Digital Mode C
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Forward step time interval
wTime1	X	
wTime2	X	
CtrlVal0	0-100	Brightness off speed
CtrlVal1	X	

### Remark

N/A.

- Digital Mode D

Name	Value	Description
LedMode	13	Digital Mode D
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Speed
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

N/A.

- Digital Mode E

Name	Value	Description
LedMode	14	Digital Mode E
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Cycle time
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

N/A.

- Digital Mode F

Name	Value	Description
LedMode	15	Digital Mode F
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	X	
wTime0	0-65535	Speed
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

N/A.

- Digital Mode G

Name	Value	Description
LedMode	16	Digital Mode G
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	Set a color RGB code
wTime0	0-65535	Forward step time interval
wTime1	X	
wTime2	X	
CtrlVal0	0-100	Brightness off speed
CtrlVal1	X	

### Remark

N/A.

- Digital Mode H

Name	Value	Description
LedMode	17	Digital Mode H
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	if no color is set, it will automatically change color
wTime0	0-65535	Speed
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

N/A.

- Digital Mode I

Name	Value	Description
LedMode	18	Digital Mode I
MaxBrightness	0-100	(%)
MinBrightness	0-100	(%)
dwColor	R/G/B	if no color is set, it will automatically change color
wTime0	0-65535	Speed
wTime1	X	
wTime2	X	
CtrlVal0	X	
CtrlVal1	X	

### Remark

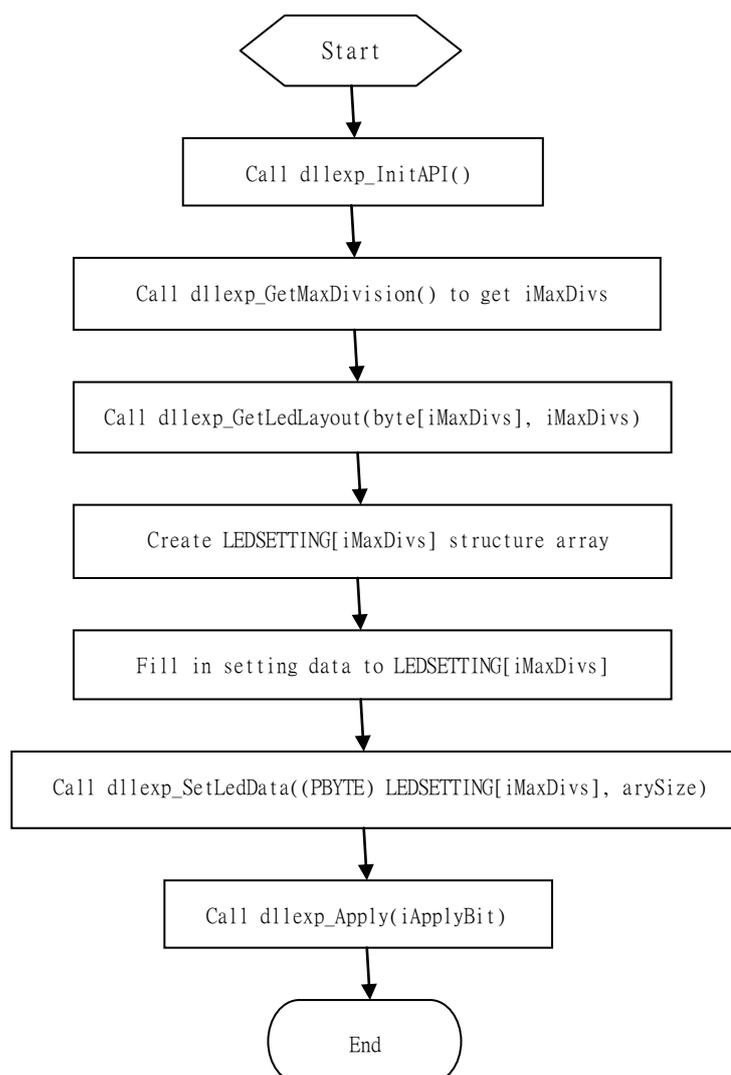
N/A.

## 4 Motherboard LED Control API Reference

The API allows the application to control the LEDs on the motherboard and the LED strip connected to the motherboard. The current LED arrangement on the motherboard (including the LED strip connected to the motherboard) can be 8 or 9-zone at maximum based on the MCU Firmware version. However, not all motherboards have the maximum 8 or 9-zone LED arrangement. When setting the data, you must set to the maximum number of LED zones the FW can support. The APIs provided by this SDK allow for individually setting different LED effects for each zone.

### 4.1 The API Use Flow

The flow chart below shows the calling sequence of controlling the LEDs on the motherboard with API.



## 4.2 APIs

This section describes the APIs export from GLedApi.dll that is used to control the LEDs on the motherboard.

### 4.2.1 dllexp\_GetSdkVersion

This method returns the revision number of the RGB Fusion SDK currently being used.

#### Syntax

```
DWORD dllexp_GetSdkVersion(LPTSTR lpBuf, int bufSize)
```

#### Parameters

Name	Type	Description
lpBuf	Output	Pointer to a buffer that receives the SDK revision string. The return string is Unicode encode.
bufSize	Input	Size, in characters, of the buffer indicated by lpBuf. This value should not be less than 16.

#### Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INSUFFICIENT_BUFFER(0x7A)	The data area passed to a function call is too small

#### Remarks

N/A.

### 4.2.2 dllexp\_InitAPI

This method initializes the SDK library on the current application.

#### Syntax

```
DWORD dllexp_InitAPI(void)
```

#### Parameters

N/A

### Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INVALID_OPERATION(0x10DD)	Fail

### Remarks

You need to initialize the SDK library before you call any of the library functions except for `dllexp_GetSdkVersion`.

### 4.2.3 `dllexp_GetMaxDivision`

This method returns the maximum number of LED zones that the MCU currently supports.

#### Syntax

```
int dllexp_GetMaxDivision(void)
```

#### Parameters

N/A.

#### Return Values

Returns the maximum number of LED zones that the MCU currently supports if success; otherwise -1 is returned.

#### Remarks

N/A.

### 4.2.4 `dllexp_GetLedLayout`

This method returns the current LED layout on the motherboard.

#### Syntax

```
DWORD dllexp_GetLedLayout(PBYTE bytArray, int arySize)
```

## Parameters

Name	Type	Description										
bytArray	Output	Pointer to a buffer that receives the LED layout on motherboard. The values of the elements in the array are as follows:										
		<table border="1"><thead><tr><th>Value</th><th>Meaning</th></tr></thead><tbody><tr><td>0(NA)</td><td>No implement LED</td></tr><tr><td>1(A_LED)</td><td>Analog LED</td></tr><tr><td>2(D_LED_TYPE1)</td><td>Digital LED Type 1</td></tr><tr><td>3(D_LED_TYPE2)</td><td>Digital LED Type 2</td></tr></tbody></table>	Value	Meaning	0(NA)	No implement LED	1(A_LED)	Analog LED	2(D_LED_TYPE1)	Digital LED Type 1	3(D_LED_TYPE2)	Digital LED Type 2
		Value	Meaning									
		0(NA)	No implement LED									
		1(A_LED)	Analog LED									
2(D_LED_TYPE1)	Digital LED Type 1											
3(D_LED_TYPE2)	Digital LED Type 2											
arySize	Input	Size, in byte of the buffer indicated by bytArray. This value must be a dllexp_GetMaxDivision return value.										

## Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success.
ERROR_INVALID_OPERATION(0x10DD)	Fail.
ERROR_INSUFFICIENT_BUFFER(0x7A)	The data area passed to a function call is too small.
ERROR_NOT_SUPPORTED(0x32)	The request is not supported.

## Remarks

If the return value is ERROR\_INVALID\_OPERATION, check to see if arySize is less than the return value of dllexp\_GetMaxDivision. If the return value is ERROR\_NOT\_SUPPORTED, it means this SDK does not support the current LED layout on the motherboard.

### 4.2.5 dllexp\_SetLedData

This method writes settings into the MCU. The settings will not apply until Dllexp\_Apply (see 4.2.6) is called to complete the whole setup process.

## Syntax

```
DWORD dllexp_SetLedData(PBYTE bytArray, int arySize)
```

### Parameters

Name	Type	Description
bytArray	Input	Pointer to a byte array converted from the LEDSETTING structure array.
arySize	Input	Size, in bytes, of the buffer indicated by bytArray.

### Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INVALID_OPERATION(0x10DD)	Fail

### Remarks

N/A.

## 4.2.6 dllexp\_Apply

This method applies settings to the MCU and the LEDs will begin to function according to the configured mode.

### Syntax

```
DWORD dllexp_Apply(int iApplyBit)
```

## Parameters

Name	Type	Description
iApplyBit	Input	This value can be used to control which LED zone to apply the new settings. You can apply the settings to all LED zones or to only a few zones. The maximum number of currently supported LED zone can be obtained by calling the <code>dllexp_GetMaxDivision</code> API (see 4.2.3). The corresponding values of each zone are as follows: 1: Apply to LED division 0 2: Apply to LED division 1 4: Apply to LED division 2 8: Apply to LED division 3 16: Apply to LED division 4 32: Apply to LED division 5 64: Apply to LED division 6 128: Apply to LED division 7 256: Apply to LED division 8 -1: Apply to all division Example: To change division 0, division 2, division 4 only, then $iApplyBit = 1 + 4 + 16 = 21$

## Return Values

Value	Description
<code>ERROR_SUCCESS(0x0)</code>	Success
<code>ERROR_INVALID_OPERATION(0x10DD)</code>	Fail

## Remarks

N/A.

### 4.2.7 `dllexp_BeatInput`

This API begins to function after the MCU enters Music mode. Programmers need to retrieve the music signals and use this API to set the LEDs to turn on or off with music signals.

## Syntax

```
DWORD dllexp_BeatInput(int iCtrl)
```

### Parameters

Name	Type	Description
iCtrl	Input	0:LED turn off, 1:LED turn on

### Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INVALID_OPERATION(0x10DD)	Fail

### Remarks

N/A.

## 4.2.8 dllexp\_IT8295\_Reset

Calling this API is equivalent to sending the last commands (including mode, speed, brightness) to the MCU again.

### Syntax

```
DWORD dllexp_IT8295_Reset(void)
```

### Parameters

N/A.

### Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INVALID_OPERATION(0x10DD)	Fail

### Remarks

N/A.

## 4.2.9 dllexp\_Get\_IT8295\_FwVer

Call this API to get the MCU firmware version.

### Syntax

```
DWORD dllexp_Get_IT8295_FwVer(PBYTE bytArray, int arySize)
```

### Parameters

Name	Type	Description
bytArray	Input	Pointer to a byte array that receives MCU firmware version.
arySize	Input	Size, in bytes, of the buffer indicated by bytArray. This value should not be less than 4.

### Return Values

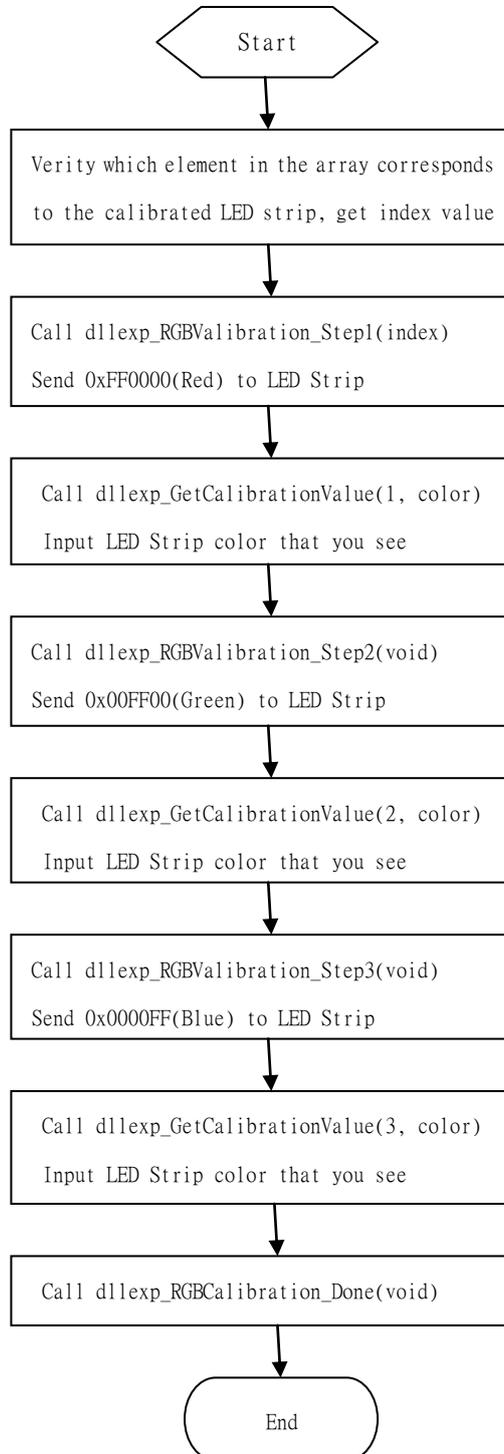
Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INVALID_OPERATION(0x10DD)	Fail
ERROR_INSUFFICIENT_BUFFER(0x7A)	The data area passed to a function call is too small.

### Remarks

Example: if the return buffer is `bytArray[] = {0x11, 0x01, 0x01, 0x0}`, then the Firmware version is 0.1.1.17.

## 4.3 RGB LED Strip Calibration

As there are various kinds of RGB LED strips on the market, the RGB pin layout of the LED strip may not be the same as that of the RGB header on the motherboard, thereby resulting in difference in output and input colors. Therefore, to accurately display colors, color calibration is necessary and the process is as follows.



### 4.3.1 dllexp\_RGBCalibration\_Step1

Calls this API to start calibrating the RGB LED strip. This function will temporarily turn off the LEDs other than the calibrated LED zone and will send 0xFF0000 (0xRRGGBB) RGB value to the calibrated LED zone. If the red pin of the LED strip corresponds to that of the header on the motherboard, the LED strip will light red.

#### Syntax

```
DWORD dllexp_RGBCalibration_Step1(int cal_div)
```

#### Parameters

The cal\_div value is the index used to correct the location of LED strip on the motherboard, that is, the previously-mentioned element index value used to control the LED strip in the LEDSETTING structure array. According to the present motherboard layout, the LED strip must be located after area 4, meaning programmers can start testing which element in the array the LED strip to be corrected is from LEDSETTING[4].

#### Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INVALID_OPERATION(0x10DD)	Fail

#### Remarks

N/A.

### 4.3.2 dllexp\_RGBCalibration\_Step2

Call the API to proceed to the step 2 of calibrating the RGB LED strip. This function will send 0x00FF00(0xRRGGBB) RGB value to the calibrated LED zone. If the green pin of the LED strip corresponds to that of the header on the motherboard, the LED strip will light green.

#### Syntax

```
void dllexp_RGBCalibration_Step2(void)
```

#### Parameters

N/A.

#### Return Values

N/A.

#### Remarks

N/A.

### 4.3.3 dllexp\_RGBCalibration\_Step3

Call the API to proceed to the step 3 of calibrating the RGB LED strip. This function will send 0x0000FF(0xRRGGBB) RGB value to the calibrated LED zone. If the blue pin of the LED strip corresponds to that of the header on the motherboard, the LED strip will light blue.

#### Syntax

```
void dllexp_RGBCalibration_Step3(void)
```

#### Parameters

N/A.

#### Return Values

N/A.

#### Remarks

N/A.

### 4.3.4 dllexp\_RGBCalibration\_Done

Call the API to finish calibrating the RGB LED strip. This function will save the correction result to the MCU and restore all LEDs to the state before the calibration.

#### Syntax

```
DWORD dllexp_RGBCalibration_Done(void)
```

#### Parameters

N/A.

#### Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success

ERROR_INVALID_OPERATION(0x10DD)	Fail
---------------------------------	------

### Remarks

N/A.

## 4.3.5 dllexp\_GetCalibrationValue

Call this API to input the actual LED strip color in step 1~3 of the calibration.

### Syntax

```
void dllexp_GetCalibrationValue(int iStep, int color_see)
```

### Parameters

Name	Type	Description
iStep	Input	Available values are 1 ~ 3, representing step 1 to step 3.
Color_see	Input	The color you see on the LED strip 1: Red 2: Green 3: Blue

### Return Values

N/A.

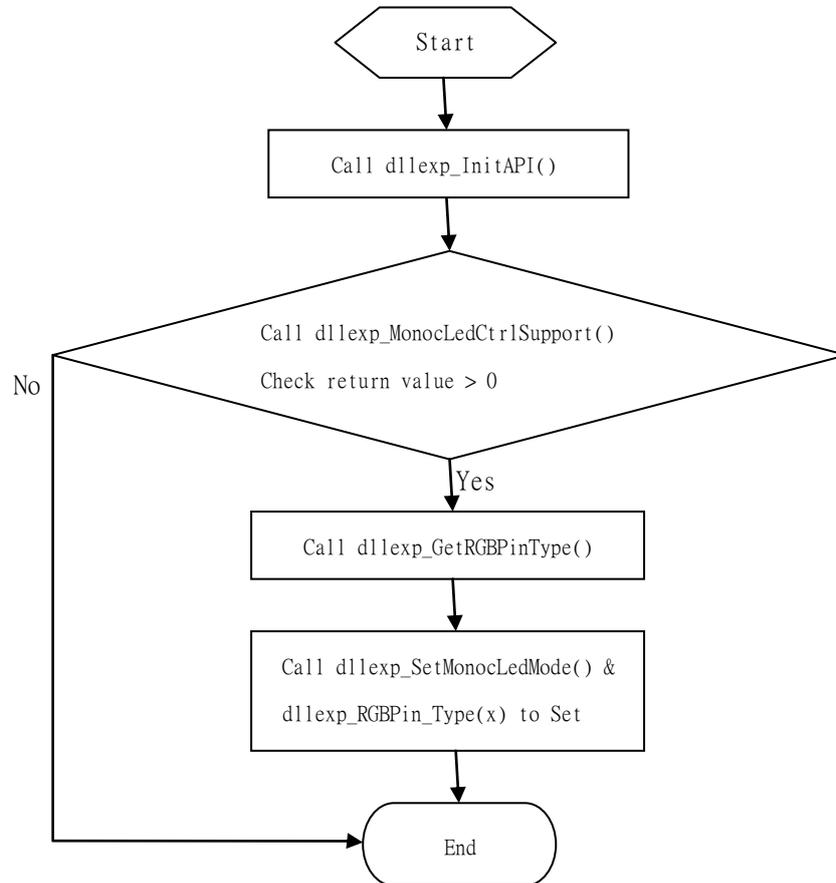
### Remarks

N/A.

## 4.4 APIs For Monochrome LED

The APIs listed in this section apply only to motherboards with monochrome LEDs. Currently, the motherboards with monochrome LEDs provide up to four basic modes: Off, Static, Pulse, and Music. You can call the API to see which modes are supported by the motherboard. Moreover, this kind of motherboards might have a LED strip header that only supports Off or Static modes while Pulse and Music modes are not supported. As to the color control, only the seven colors generated by the RGB switch are available, not full colors.

#### 4.4.1 Monochrome LED APIs Flow



#### 4.4.2 dllexp\_MonocLedCtrlSupport

Call the API to verify if monochrome LED control is supported. If the return value is greater than 0, it means monochrome LED control is supported. Moreover, you can see which monochrome modes are supported by analyzing the return value.

##### Syntax

```
int dllexp_MonocLedCtrlSupport(void)
```

##### Parameters

N/A.

### Return Values

Value	Description
0	Not support
1	Support static mode
2	Support pulse mode
4	Support music mode

The return value can be generated by combining the three values, 1, 2, 4. For example, if the return value is 7, it means the three modes above are supported.

### Remarks

To achieve monochrome LED music mode, programmers also have to capture music signals and then call `dllexp_BeatInput`(see 4.2.7).

### 4.4.3 `dllexp_GetRGBPinType`

Call the API to decide which function to call to control the LED strip. If the the return value is x, it means you need to call the `dllexp_RGBPin_Type (x)` function to control the LED strip.

#### Syntax

```
int dllexp_GetRGBPinType(void)
```

#### Parameters

N/A.

### Return Values

Value	Description
0	Not support
1	Call <code>dllexp_RGBPin_Type1</code> to control RGB LED Strip

### Remarks

N/A.

### 4.4.4 `dllexp_SetMonocLedMode`

Call the API to control the LED modes of a motherboard with monochrome LEDs. Currently supported modes are Off mode, Static mode, Pulse Mode, and Music mode.

## Syntax

```
BOOL dllexp_SetMonocLedMode(int mnoLedMode)
```

## Parameters

Name	Type	Description
mnoLedMode	Input	Available value are as follows: 0: Off mode 1: Static mode 2: Pulse mode 3: Music mode

## Return Values

Return TRUE, if success else return FALSE.

## Remarks

N/A.

### 4.4.5 dllexp\_RGBPin\_Type1

Call the API to control the RGB LED strip color of a motherboard with monochrome LEDs. The three parameters respectively control the three pins of the RGB LED strip header on the motherboard. Accepted values are 0 (off) and 1 (on).

## Syntax

```
DWORD dllexp_RGBPin_Type1(int pin1, int pin2, int pin3)
```

## Parameters

Name	Type	Description
pin1/pin2/pin3	Input	Available values are as follows: 0: R/G/B light turn off 1: R/G/B light turn on

## Return Values

Value	Description
ERROR_SUCCESS(0x0)	Success
ERROR_INVALID_OPERATION(0x10DD)	Fail

## Remarks

As the pin layout of the RGB LED strips on the market may not be exactly the

same, programmers have to write their own RGB correction codes. The purpose of the correction codes is to know how pins 1, 2, 3 of the motherboard LED strip header correspond to the R, G, B pins of the LED strip in order to correctly control the R, G, B values when configuring the color.

## 5 VGA & Peripherals LED Control API Reference

The API allows the application to control the LEDs on the VGA and peripherals LED connected to the motherboard.

### 5.1 APIs

This section describes the APIs export from GvLedLib.dll that is used to control the LEDs on the VGA & peripherals.

#### 5.1.1 dllexp\_GvLedInitial

This method initializes the SDK library on the current application.

##### Syntax

```
DWORD dllexp_GvLedInitial(int &iDeviceCount, int *iDeviceIdArray)
```

## Parameters

Name	Type	Description																																		
iDeviceCount	Output	The count connected to PC that LED can be controlled.																																		
iDeviceIdArray	Output	Pointer to a buffer that receives the LED connected to PC. The values of the elements in the array are as follows: <table border="1" data-bbox="735 685 1355 1518"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0x1001</td> <td>VGA</td> </tr> <tr> <td>0x2001</td> <td>XK700 Keyboard</td> </tr> <tr> <td>0x2002</td> <td>AORUS K7 Keyboard</td> </tr> <tr> <td>0x2003</td> <td>AORUS K9 Keyboard</td> </tr> <tr> <td>0x3001</td> <td>XM300 Mouse</td> </tr> <tr> <td>0x3002</td> <td>AORUS M3 Mouse</td> </tr> <tr> <td>0x4001</td> <td>1070IXEB Gaming VGA Box</td> </tr> <tr> <td>0x4002</td> <td>1080IXEB Gaming VGA Box</td> </tr> <tr> <td>0x4003</td> <td>XTC700 Case</td> </tr> <tr> <td>0x4004</td> <td>XC300W Case</td> </tr> <tr> <td>0x4005</td> <td>XC700W Case</td> </tr> <tr> <td>0x4006</td> <td>XH300 Earphone</td> </tr> <tr> <td>0x4007</td> <td>AORUS H5 Earphone</td> </tr> <tr> <td>0x4008</td> <td>AC300W Case</td> </tr> <tr> <td>0x4009</td> <td>ATC700 CPU Cooler</td> </tr> <tr> <td>0x400A</td> <td>AORUS P7 Mouse pad</td> </tr> </tbody> </table>	Value	Meaning	0x1001	VGA	0x2001	XK700 Keyboard	0x2002	AORUS K7 Keyboard	0x2003	AORUS K9 Keyboard	0x3001	XM300 Mouse	0x3002	AORUS M3 Mouse	0x4001	1070IXEB Gaming VGA Box	0x4002	1080IXEB Gaming VGA Box	0x4003	XTC700 Case	0x4004	XC300W Case	0x4005	XC700W Case	0x4006	XH300 Earphone	0x4007	AORUS H5 Earphone	0x4008	AC300W Case	0x4009	ATC700 CPU Cooler	0x400A	AORUS P7 Mouse pad
Value	Meaning																																			
0x1001	VGA																																			
0x2001	XK700 Keyboard																																			
0x2002	AORUS K7 Keyboard																																			
0x2003	AORUS K9 Keyboard																																			
0x3001	XM300 Mouse																																			
0x3002	AORUS M3 Mouse																																			
0x4001	1070IXEB Gaming VGA Box																																			
0x4002	1080IXEB Gaming VGA Box																																			
0x4003	XTC700 Case																																			
0x4004	XC300W Case																																			
0x4005	XC700W Case																																			
0x4006	XH300 Earphone																																			
0x4007	AORUS H5 Earphone																																			
0x4008	AC300W Case																																			
0x4009	ATC700 CPU Cooler																																			
0x400A	AORUS P7 Mouse pad																																			

## Return Values

Value	Description
GV_LED_API_OK (0x0)	Success

## Remarks

You need to initialize the SDK library before you call any of the library functions.

### 5.1.2 dllexp\_GvLedGetVersion

This method returns the revision number of the GvLedLib.dll currently being used.

#### Syntax

```
DWORD dllexp_GetSdkVersion(int &iMajorVersion, int &iMinorVersion)
```

#### Parameters

Name	Type	Description
iMajorVersion	Output	Major version of GvLedLib.dll.
iMinorVersion	Output	Minor version of GvLedLib.dll.

#### Return Values

Value	Description
GV_LED_API_OK (0x0)	Success

#### Remarks

N/A.

### 5.1.3 dllexp\_GvLedSet

This method writes LED settings into the devices.

#### Syntax

```
DWORD dllexp_GvSetLed(int nIndex, GVLED_CFG config)
```

#### Parameters

Name	Type	Description
nIndex	Input	The device id want to apply the LED setting, reference 6.1.1 iDeviceIdArray setting -1 means all devices
Config	Input	The LED effect setting, please reference 5.1 LED Setting Data structure.

### Return Values

Value	Description
GV_LED_API_OK (0x0)	Success
GV_LED_API_DEVICE_NOT_AVAILABLE (0x2)	Fail, the device is not available.
GV_LED_API_ERROR_PARAM(3)	Fail, parameter error.

### Remarks

N/A.

## 5.1.4 dllexp\_GvLedGetVgaModelName

This method gets the VGA model name if VGA is supported LED control.

### Syntax

```
DWORD dllexp_GvLedGetVgaModelName(byte *pVgaModelName)
```

### Parameters

Name	Type	Description
pVgaModelName	Output	The character array that contains the VGA model name.

### Return Values

Value	Description
GV_LED_API_OK (0x0)	Success

### Remarks

N/A.