Programming Guide
for
SDA Lighting Controller

Revision 1.0.1 Nov, 2016

Revision History

Revision	***Date***	***Description***
1.0.0	2015-07-24	Initial version.
1.0.1	2016-11-10	added new features - for dll ver 6.0.2 or later - added LEDShutdownTemperature - added Controller Status “DETECTED LED CURRENT OUT OF RANGE”

Technical Support

Tel : +604-646 8428

Email : sales@tms-lite.com

Website : www.tms-lite.com

Table of Contents

Chapter 1. Overview

Two DLLs, one for Win32 applications and one for .NET applications, are provided for Windows programming. The APIs are designed for simplicity and programmers should be easily to be familiar with the APIs.

1.1 System Requirements

Supported Operating System

- Windows 8
- Windows 7
- Windows Vista with Service Pack 2
- Windows XP with Service Pack 3

Prerequisites:

- Users will need to install VS2010 redistributable if not already present
- .NET Framework: You must install either the .NET Framework 3.5, .NET Framework 4, or .NET Framework 4.5 on every end user computer if you use the .NET dll.

Chapter 2. Dynamic Link Libraries (DLLs)

2.1 Using of Win32 DLLs in Visual Studio

SDA APIs for Win32 are written in C. In order to link with SDA APIs provided in the DLL, the ZKA32.lib file has to be added to the project or specified in the linker's options. The function prototype of the APIs are provided in the CBridgeSDA.h file which has to be included in the .cpp file in which the APIs are called.

2.2 Using of .NET DLL in Visual Studio

SDA APIs for .NET are encapsulated in a class "SDA_Driver" contained in the namespace "ZKA". In order to use the DLL, it is necessary to add the ZKANet.dll to the project's references (Project->Add Reference->Browse->ZKANet.dll)

Documentation of the library are located in the ZKANet.XML file. In order to display the documentation of the API in the Visual Studio's editor, the ZKANet.XML has to be put in the same directory as ZKANet.dll.

All class methods are provided in the ZKANet.dll. All of the lighting controller functions are encapsulated in the class "SDA_Driver". Additionally, a high resolution timer class "HiResTimer" is also provided for time measurement.

"SDA_Driver" and "HiResTimer" classes are encapsulated in the namespace ZKA. It is recommended to add the "using ZKA" directive in the code.

e.g.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;

using ZKA; // must have this line

namespace SDADemo
{
  public partial class FormSDADemo : Form
  {
        ...
  }
  ...
}
```

Chapter 3. Application Programming Interface (API)

3.1 Member Functions

3.1.1 Device Initialization

---

1 Constructor

Type : Device Initialization
C Syntax : ***SDA_Driver* CreateSDA_Driver ()***
C# Syntax : ***SDA_Driver ()***
Description : Constructor of SDA_Driver
Parameters : pointer to the SDA_Driver object (C only)
Return Value : -

---

2 OpenDevice

Type : Device Initialization
C Syntax : ***BOOL SDA_M_OpenDevice (SDA_Driver* sda_driver, char* ipaddr="")***
C# Syntax : ***bool OpenDevice (string ipaddr)***
Description : Open device.
Parameters : ipaddr – IP address of the SDA master.
Return Value : true – open device succeed, false – open device fail

---

3 Init

Type : Device Initialization
C Syntax : ***BOOL SDA_M_Init (SDA_Driver* sda_driver, unsigned __int32 channel_id)***
C# Syntax : ***bool Init (UInt32 channel_id)***
Description : Initialize the driver.
Parameters : channel_id – Channel ID of the SDA controller.
Return Value : true – command succeeds, false – command fails

---

4 CloseDevice

Type : Device Initialization
C Syntax : ***void SDA_M_CloseDevice (SDA_Driver* sda_driver)***
C# Syntax : ***void CloseDevice ()***
Description : Close device.

C Example :

```
//Create SDA_Driver object.
SDA_Driver* sda_driver = CreateSDA_Driver ();

// open and connect to the SDA master with IP address = 192.168.1.100,
// if failed, display a message and exit the function
char ip[] = "192.168.1.100";
if (!SDA_M_OpenDevice (sda_driver, ip){
   MessageBox ("Open device failed!");
   return;
}

// initialize channel 1, if failed, display a message and exit the
// function
unsigned int channel_id = 1;
if (!SDA_M_Init (sda_driver, channel_id)){
   MessageBox ("Initialize channel 1 failed!");
   return;
}

//control the LED light
…

// close device
if (sda_driver != NULL){
   SDA_M_CloseDevice (sda_driver);
   delete sda_driver;
   sda_driver = NULL;
}
```

C# Example :

```
//Create SDA_Driver object.
SDA_Driver sda_driver;
sda_driver = new SDA_Driver ();

// open and connect to the SDA master with IP address = 192.168.1.100,
// if failed, display a message and exit the function
string s = "192.168.1.100";
IPAddress ip;
if (IPAddress.TryParse (s, out ip)) // check if ip is valid
{
   if (sda_driver.OpenDevice (s)){
      //open device success
   } else
      MessageBox.Show ("Open Device Fail!", "Error!",
                       MessageBoxButtons.OK, MessageBoxIcon.Error);
} else {
   MessageBox.Show ("Invalid IP!", "Error!", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
}

//initialize channel 1, if failed, display a message and exit the //funtion.
uint channel_id = 1;
if (!sda_driver.Init (channel_id))
```

```
{
   MessageBox.Show ("Initialize channel 1 fail!", "Error!",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
   return;
}

//control the LED light
...

// close device
if (sda_driver != null)
{
   sda_driver.CloseDevice ();
   sda_driver = null;
}
```

3.1.2 DLL & Controller Information

1 GetDLLProgramVersion

Type : DLL & Controller Information
C Syntax : ***void SDA_M_GetDLLProgramVersion (***
SDA_Driver* sda_driver, __int32 &major_version,
__int32 &minor_version, __int32 &revision)
C# Syntax : ***static void GetDLLProgramVersion (***
out Int32 major_version, out Int32 minor_version,
out Int32 revision)
Description : Get the version number of the DLL.
Parameters : channel_id – Channel ID of the SDA controller.
version1 – Major version.
version2 – Minor version.
version3 – Revision.
Return Value : -

C Example :

```
// read and display the version number of the DLL
int version[3];
SDA_M_GetDLLProgramVersion (sda_driver, version[0], version[1], version[2]);

char c_version[3][256];
for (int i=0; i<3; i++)
{
   _itoa_s (version[i], c_version[i], 10);
}
MessageBox ("DLL Version: "+c_version[0] + "." + c_version[1] + "."
        +c_version[2]);
```

C# Example :

```
//read and display the version number of the DLL.
int version1, version2, version3;

sda_driver.GetDLLProgramVersion (out version1, out version2, out version3);

string dll_version = (version1).ToString () + "." +
                     (version2).ToString () + "." +
                     (version3).ToString ();

MessageBox.Show ("DLL Version: "+dll_version, "DLL Version",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
```

2 GetE2CConverterFirmwareVersion

Type : DLL & Controller Information
C Syntax : ***void SDA_M_GetE2CConverterFirmwareVersion (SDA_Driver* sda_driver, unsigned __int32 &major_version, unsigned __int32 &minor_version, unsigned __int32 &revision)***
C# Syntax : ***void GetE2CConverterFirmwareVersion (out UInt32 major_version, out UInt32 minor_version, out UInt32 revision)***
Description : Get the firmware version of the Ethernet to CAN (SDA Master) converter
Parameters : channel_id – Channel ID of the SDA controller.
version1 – Major version.
version2 – Minor version.
version3 – Revision.
Return Value : true – command succeeds, false – command fails

C Example :

```
// read and display the firmware version number of the Ethernet to
// CAN converter
unsigned int version[3];
if (!SDA_M_GetE2CConverterFirmwareVersion (sda_driver, version[0],
version[1],
                                           version[2]))
{
   MessageBox ("Can't read program version of the Ethernet to CAN
               (SDA Master) converter");
}
else
{
   char c_version[3][256];
   for (int i=0; i<3; i++)
   {
      _itoa_s (version[i], c_version[i], 10);
   }
   MessageBox ("Firmware Version of E2C :" + c_version[0] + "." +
              c_version[1] + "." +c_version[2]);
}
```

C# Example :

```
// read and display the firmware version number of the Ethernet to
// CAN (SDA Master) converter
int version1, version2, version3;
if (!sda_driver.GetE2CConverterFirmwareVersion (out version1,
                                                out version2,
                                                out version3))
{
   MessageBox.Show ("Can't read program version of the Ethernet to
                    CAN converter!", "Error!", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
}
else
{
   string e2c_version = (version1).ToString () + "." +
                        (version2).ToString () + "." +
                        (version3).ToString ();
```

```
   MessageBox.Show ("E2C Version: "+e2c_version, "E2C Version",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

---

3 GetControllerFirmwareVersion

Type : DLL & Controller Information
C Syntax : ***BOOL SDA_M_GetControllerFirmwareVersion (***
SDA_Driver* sda_driver, unsigned __int32 channel_id,
unsigned __int32 &major_version,
unsigned __int32 &minor_version, unsigned __int32 &revision)
C# Syntax : ***bool GetControllerFirmwareVersion (UInt32 channel_id,***
out UInt32 major_version, out UInt32 minor_version,
out UInt32 revision)
Description : get the firmware version of the SDA controller.
Parameters : channel_id – Channel ID of the SDA controller.
version1 – Major version.
version2 – Minor version.
version3 – Revision.
Return Value : true – command succeeds, false – command fails

C Example :

```
// read and display the firmware version number of the SDA
// controller
unsigned int channel_id = 1;
unsigned int version[3];
if (SDA_M_GetControllerFirmwareVersion (sda_driver, channel_id, version[0],
                                        version[1], version[2]))
{
   char c_version[3][256];
   for (int i=0; i<3; i++)
   {
      _itoa_s (version[i], c_version[i], 10);
   }
   MessageBox ("Program Version : " + c_version[0] + "." + c_version[1]
              + "." +c_version[2]);
}
else
{
   MessageBox ("Connection fail!");
   return;
}
```

C# Example :

```
// read and display the firmware version number of the SDA
// controller
uint channel_id = 1;
uint version1, version2, version3;
if (sda_driver.GetControllerFirmwareVersion (channel_id, out version1,
                                        out version2, out
                                    version3)
{
   string controller_version = (version1).ToString () + "." +
                               (version2).ToString () + "." +
                               (version3).ToString ();

   MessageBox.Show ("Controller firmware Version: "+controller_version,
                    "E2C Version", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
}
else
{
   MessageBox.Show ("Connection fail!", "Error!", MessageBoxButtons.OK,
                     MessageBoxIcon.Error);
   return;
}
```

---

4 GetControllerModelName

Type : DLL & Controller Information
C Syntax : ***BOOL SDA_M_GetControllerModelName (SDA_Driver* sda_driver, unsigned __int32 channel_id, unsigned __int64 &model_name)***
C# Syntax : ***bool GetControllerModelName (UInt32 channel_id, out string model_name)***
Description : Get the model name of the SDA_Driver
Parameters : channel_id – Channel ID of the SDA Driver.
Return Value : true – command succeeds, false – command fails

C Example :

```
// get the model name of the driver with channel id = 1
unsigned int channel_id = 1;
unsigned __int64 u64_model_name;
if (SDA_M_GetControllerModelName (sda_driver, channel_id, u64_model_name)
{
   char name1 = (char) ( (u64_model_name >> 32) & 0xff);
   char name2 = (char) ( (u64_model_name >> 24) & 0xff);
   char name3 = (char) ( (u64_model_name >> 16) & 0xff);
   char name4 = (char) ( (u64_model_name >> 8) & 0xff);
   char name5 = (char) ( (u64_model_name >> 0) & 0xff);

   MessageBox ("Model name of the driver : " + name1 + name2 + name3 + name4
               + name5);
}
else
{
```

```
   MessageBox ("Fail to get the model name!");
   return;
}
```

C Example :

```
// get the model name of the driver with channel id = 1
uint channel_id = 1;
string model_name;
if (sda_driver.GetControllerModelName (ui_channel_id, out model_name)
{
   MessageBox.Show ("Driver Model: "+ model_name,
                    "Driver Model", MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
}
else
{
   MessageBox.Show ("Connection fail!", "Error!", MessageBoxButtons.OK,
                     MessageBoxIcon.Error);
   return;
}
```

3.1.3 Get/Set Parameters

1 SetOperationMode / GetOperationMode

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetOperationMode (SDA_Driver* sda_driver, unsigned __int32 channel_id, unsigned __int32 operation_mode)***
BOOL SDA_M_GetOperationMode (SDA_Driver* sda_driver, unsigned __int32 channel_id, unsigned __int32 &operation_mode)
C# Syntax : ***bool SetOperationMode (UInt32 channel_id, UInt32 operation_mode)***
bool GetOperationMode (UInt32 channel_id, out UInt32 operation_mode)
Description : *Set/Get the operation mode of the SDA_Driver*
Parameters : *channel_id – Channel ID of the SDA Driver*
operation_mode, 0 - Continuous Mode, 1 - Pulse Mode
Return Value : *true – command succeeds, false – command fails*

2 SetRatedType / GetRatedType

(NOTE: Only Voltage Rated Type is supported in standard version)
Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetRatedType (SDA_Driver* sda_driver, unsigned __int32 channel_id, unsigned __int32 rated_type)***
BOOL SDA_M_GetRatedType (SDA_Driver* sda_driver, unsigned __int32 channel_id, unsigned __int32 &rated_type)
C# Syntax : ***bool SetRatedType (UInt32 channel_id, UInt32 rated_type)***
bool GetRatedType (UInt32 channel_id, out UInt32 rated_type)
Description : Set/Get the Rated Type of the SDA_Driver
Parameters : channel_id – Channel ID of the SDA Driver
rated_type, 0 – Current Rated Type, 1 – Voltage
Return Value : *true – command succeeds, false – command fails*

3 SetRatedCurrent / GetRatedCurrent

(NOTE: Only Voltage Rated Type is supported in standard version)
Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetRatedCurrent (SDA_Driver* sda_driver, unsigned __int32 board_no, float current_A)***
BOOL SDA_M_GetRatedCurrent (SDA_Driver* sda_driver, unsigned __int32 board_no, float &current_A)
C# Syntax : ***bool SetRatedCurrent (UInt32 board_id, float current_A)***
bool GetRatedCurrent (UInt32 board_id, out float current_A)
Description : Set/Get the rated current of the SDA_Driver
Parameters : channel_id – Channel ID of the SDA Driver
current_A – rated current (unit: A)
Return Value : *true – command succeeds, false – command fails*

4 SetRatedVoltage / GetRatedVoltage

(NOTE: Rated Voltage is fixed to 24V in standard version)
Type : Driver Operation

C Syntax : ***BOOL SDA_M_SetRatedVoltage (SDA_Driver* sda_driver, unsigned __int32 board_no, float voltage_V)***
BOOL SDA_M_GetRatedVoltage (SDA_Driver* sda_driver, unsigned __int32 board_no, float &voltage_V)
C# Syntax : ***bool SetRatedVoltage (UInt32 board_id, float voltage_V)***
bool GetRatedVoltage (UInt32 board_id, out float voltage_V)
Description : Set/Get the rated voltage of the SDA_Driver
Parameters : channel_id – Channel ID of the SDA Driver
voltage_V – rated voltage (unit: V)
Return Value : true – command succeeds, false – command fails

---

5 SetContinuousModeIntensity / GetContinuousModeIntensity

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetContinuousModeIntensity (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 intensity)***
BOOL SDA_M_GetContinuousModeIntensity (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 &intensity)
C# Syntax : ***bool SetContinuousModeIntensity (UInt32 board_id, UInt32 intensity)***
bool GetContinuousModeIntensity (UInt32 board_id, out UInt32 intensity)
Description : Set/Get the Intensity (Percentage of Full Power) of Continuous Mode
Parameters : channel_id – Channel ID of the SDA Driver
intensity – intensity (unit: %)
Return Value : true – command succeeds, false – command fails

---

6 SetPulseModeOverDrivePercentage / GetPulseModeOverDrivePercentage

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetPulseModeOverDrivePercentage (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 percentage)***
BOOL SDA_M_SDA_M_GetPulseModeOverDrivePercentage (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 &percentage)
C# Syntax : ***bool SetPulseModeOverDrivePercentage (UInt32 board_id, UInt32 percentage)***
bool GetPulseModeOverDrivePercentage (UInt32 board_id, out UInt32 percentage)
Description : Set/Get the Overdirve Percentage of Pulse Mode
Parameters : channel_id – Channel ID of the SDA Driver
operation_mode, 0 - Continuous Mode, 1 - Pulse Mode
Return Value : true – command succeeds, false – command fails

---

7 SetPulseModeTriggerDelay / GetPulseModeTriggerDelay

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetPulseModeTriggerDelay (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 delay_us)***
BOOL SDA_M_GetPulseModeTriggerDelay (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 &delay_us)
C# Syntax : ***bool SetPulseModeTriggerDelay (UInt32 board_id, UInt32 delay_us)***
bool GetPulseModeTriggerDelay (UInt32 board_id, out UInt32 delay_us)
Description : Set/Get the Trigger Delay of Pulse Mode
Parameters : channel_id – Channel ID of the SDA Driver

delay_us – trigger delay of pulse mode (unit: us)
Return Value : *true – command succeeds, false – command fails*

---

8 SetPulseModePulseWidth / GetPulseModePulseWidth

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetPulseModePulseWidth (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 pulse_width_us)***
BOOL SDA_M_SetPulseModePulseWidth (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 pulse_width_us)
C# Syntax : ***bool SetPulseModePulseWidth (UInt32 board_id, UInt32 pulse_wdith_us)***
bool GetPulseModePulseWidth (UInt32 board_id, out UInt32 pulse_wdith_us)
Description : Set/Get the Pulse Width of the Pulse Mode
Parameters : channel_id – Channel ID of the SDA Driver
pulse_wdith_us – pulse width of pulse mode (unit: us)
Return Value : *true – command succeeds, false – command fails*

---

9 SetTriggerPolarity / GetTriggerPolarity

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetTriggerPolarity (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 trigger_polarity)***
BOOL SDA_M_GetTriggerPolarity (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 &trigger_polarity)
C# Syntax : ***bool SetTriggerPolarity (UInt32 board_id, UInt32 trigger_polarity)***
bool GetTriggerPolarity (UInt32 board_id, out UInt32 trigger_polarity)
Description : Set/Get the Pulse Mode Trigger Polarity
Parameters : channel_id – Channel ID of the SDA Driver
trigger_polarity – 0: Falling Edge Trigger, 1: Rising Edge Trigger
Return Value : *true – command succeeds, false – command fails*

---

10 SetConstantVoltageModeEnabled / GetConstantVoltageModeEnabled

(NOTE: Constant Voltage Mode is not supported in standard version)
Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetConstantVoltageModeEnabled (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 enabled)***
BOOL SDA_M_GetConstantVoltageModeEnabled (SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 &enabled)
C# Syntax : ***bool SetConstantVoltageModeEnabled (UInt32 board_id, UInt32 enabled)***
bool GetConstantVoltageModeEnabled (UInt32 board_id, out UInt32 enabled)
Description : Enabled/Disable the Constant Votlage Mode
Parameters : channel_id – Channel ID of the SDA Driver
enabled – 0: Disabled, 1: Enabled
Return Value : *true – command succeeds, false – command fails*

---

11 SetConstantContinuousVoltage / GetConstantContinuousVoltage

(NOTE: Constant Voltage Mode is not supported in standard version)

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetConstantContinuousVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float voltage_V)***
BOOL SDA_M_GetConstantContinuousVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float &voltage_V)
C# Syntax : ***bool SetConstantContinuousVoltage(UInt32 board_id, float voltage_V)***
bool GetConstantContinuousVoltage(UInt32 board_id, out float voltage_V)
Description : Set/Get the Voltage of the Constant Voltage Mode
Parameters : channel_id – Channel ID of the SDA Driver
voltage_V – continuous voltage of constant voltage mode(unit: V)
Return Value : true – command succeeds, false – command fails

---

12 SetConstantPulseVoltage / GetConstantPulseVoltage

(NOTE: Constant Voltage Mode is not supported in standard version)

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetConstantPulseVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float voltage_V)***
BOOL SDA_M_GetConstantPulseVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float &voltage_V)
C# Syntax : ***bool SetConstantPulseVoltage(UInt32 board_id, float voltage_V)***
bool GetConstantPulseVoltage(UInt32 board_id, out float voltage_V)
Description : Set/Get the Pulse Voltage of the Constant Voltage Mode
Parameters : channel_id – Channel ID of the SDA Driver
voltage_V – pulse voltage of constant voltage mode(unit: V)
Return Value : true – command succeeds, false – command fails

---

13 SetCurrentForRatedVoltage / GetCurrentForRatedVoltage

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetCurrentForRatedVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float current_for_rated_voltage_A)***
BOOL SDA_M_GetCurrentForRatedVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float &current_for_rated_voltage_A)
C# Syntax : ***bool SetCurrentForRatedVoltage(UInt32 board_id, float current_A)***
bool GetCurrentForRatedVoltage(UInt32 board_id, out float current_A)
Description : Set/Get the actual output of the rated voltage type in constant current mode
Parameters : channel_id – Channel ID of the SDA Driver
current_A – actual output current (unit: A)
Return Value : true – command succeeds, false – command fails

---

14 SetLEDShutdownTemperature / GetLEDShutdownTemperature

Type : *Parameter Function*
C Syntax : ***BOOL SDA_M_SetLEDShutdownTemperature(SDA_Driver* sda_driver,***
unsigned __int32 board_no,
unsigned __int32 shutdown_temperature_C)
BOOL SDA_M_GetCurrentForRatedVoltage(SDA_Driver* sda_driver,
unsigned __int32 board_no,
unsigned __int32 &shutdown_temperature_C)
C# Syntax : ***bool SetLEDShutdownTemperature (***
UInt32 board_id,
UInt32 led_shutdown_temperature_C)
bool GetLEDShutdownTemperature (
UInt32 board_id,
out UInt32 led_shutdown_temperature_C)
Description : Set/Get the LED Shutdown Temperature of LED lamp
Parameters : channel_id – Channel ID of the SDA Driver
led_shutdown_temperature_C – shutdown temperature of the LED lamp
(unit: Degrees Celsius)
Return Value : true – command succeeds, false – command fails

3.1.4 Operation Information

1 GetSupplyVoltage

Type : Operation Informaton
C Syntax : ***BOOL SDA_M_GetSupplyVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float &voltage_V)***
C# Syntax : ***bool GetSupplyVoltage(UInt32 board_id, out float voltage_V)***
Description : Get the value of input supply voltage
Parameters : channel_id – Channel ID of the SDA Driver
voltage_V – measured supply voltage (unit: V)
Return Value : true – command succeeds, false – command fails

2 GetLEDContinuousModeVoltage

(NOTE: For Rated Current Type only, will always return 24v in standard version)
Type : Operation Informaton
C Syntax : ***BOOL SDA_SDA_M_GetLEDContinuousModeVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float &voltage_V)***
C# Syntax : ***bool GetLEDContinuousModeVoltage(UInt32 board_id, out float voltage_V)***
Description : Get the measured voltage of Rated Current Type LED
Parameters : channel_id – Channel ID of the SDA Driver
voltage_V – measured voltage (unit: V)
Return Value : true – command succeeds, false – command fails

3 GetLEDContinuousModeCurrent

Type : Operation Informaton
C Syntax : ***BOOL SDA_M_GetLEDContinuousModeCurrent(SDA_Driver* sda_driver, unsigned __int32 board_no, float &current_A)***
C# Syntax : ***bool GetLEDContinuousModeCurrent(UInt32 board_id, out float current_A)***
Description : Get the measured current of Rated Voltage Type LED
Parameters : channel_id – Channel ID of the SDA Driver
current_A – measured current (unit: A)
Return Value : true – command succeeds, false – command fails

4 GetLEDPulseModeVoltage

Type : Operation Informaton
C Syntax : ***BOOL SDA_M_GetLEDPulseModeVoltage(SDA_Driver* sda_driver, unsigned __int32 board_no, float &voltage_V)***
C# Syntax : ***bool GetLEDPulseModeVoltage(UInt32 board_id, out float voltage_V)***
Description : Get the pulse mode voltage of the LED
Parameters : channel_id – Channel ID of the SDA Driver
voltage_V – pulse mode voltage (unit: V)
Return Value : true – command succeeds, false – command fails

5 GetLEDPulseModeCurrent

Type : Operation Informaton
C Syntax : ***BOOL SDA_M_GetLEDPulseModeCurrent(SDA_Driver* sda_driver, unsigned __int32 board_no, float &current_A)***
C# Syntax : ***bool GetLEDPulseModeCurrent(UInt32 board_id, out float current_A)***
Description : Get the pulse mode current of the LED
Parameters : channel_id – Channel ID of the SDA Driver
current_A – pulse mode current (unit: A)
Return Value : true – command succeeds, false – command fails

---

6 GetLEDTemperature

Type : Operation Informaton
C Syntax : ***BOOL SDA_M_GetLEDTemperature(SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int8 &temperature_C)***
C# Syntax : ***bool GetLEDTemperature(UInt32 board_id, out UInt16 Temperature_C)***
Description : Get the measured temperature of the LED
Parameters : channel_id – Channel ID of the SDA Driver
temperature_C – temperature of the LED in degree celsius
Return Value : true – command succeeds, false – command fails

---

7 GetControllerStatus

Type : Operation Informaton
C Syntax : ***BOOL SDA_M_GetControllerStatus(SDA_Driver* sda_driver, unsigned __int32 board_no, unsigned __int32 &status)***
C# Syntax : ***bool GetControllerStatus(UInt32 board_id, out UInt32 status)***
Description : Get the status of the controller
Parameters : channel_id – Channel ID of the SDA Driver
status: 0 – RESTART,
1 – NORMAL,
2 – OVERCURRENT,
3 – OPEN_CIRCUIT,
4 – REQUIRED_VOLTAGE_TOO_HIGH,
5 – RATED_CURRENT_TOO_LOW,
6 – CURRENT_SETTING_TOO_HIGH,
7 – OVER_TEMPERATURE
8 – DETECTED LED CURRENT OUT OF RANGE
Return Value : true – command succeeds, false – command fails

3.1.5 Operation Commands

1 Trigger

Type : Driver Operation
C Syntax : ***BOOL SDA_M_Trigger(SDA_Driver* sda_driver, unsigned __int32 board_no)***
C# Syntax : ***bool Trigger(UInt32 board_id)***
Description : Start Software Trigger (General a pulse in pulse mode)
Parameters : channel_id – Channel ID of the SDA Driver
Return Value : true – command succeeds, false – command fails

2 SaveParamToFlash

Type : Driver Operation
C Syntax : ***BOOL SDA_M_SaveParamToFlash(SDA_Driver* sda_driver, unsigned __int32 board_no)***
C# Syntax : ***bool SaveParamToFlash(UInt32 board_id)***
Description : Save the current parameters to the controller
Parameters : channel_id – Channel ID of the SDA Driver
Return Value : true – command succeeds, false – command fails

3 Restart

Type : Driver Operation
C Syntax : ***BOOL SDA_M_Restart(SDA_Driver* sda_driver, unsigned __int32 board_no)***
C# Syntax : ***bool Restart(UInt32 board_id)***
Description : Restart the LED measure process
Parameters : channel_id – Channel ID of the SDA Driver
Return Value : true – command succeeds, false – command fails