Programming Manual for OPT Digital Light Source Controller

Version 1.0.15

Contents

Revising Record

Version	Date	Remarks
1.0.0	17-09-2014	Developed by OPT MACHINE VISION TECH CO., LTD
1.0.1	14-11-2014	Developed by OPT MACHINE VISION TECH CO., LTD
1.0.5	19-11-2014	Developed by OPT MACHINE VISION TECH CO., LTD
1.0.6	03-12-2014	Fixing Bugs (for details see README.txt)
1.0.7	07-03-2015	Add function OPTController_ConnectionResetBySN
1.0.8	09-03-2015	Add functions OPTController_ConnectionResetByIP and OPTController_SetEtheConnectionHeartBeat
1.0.9	20-03-2015	Improving heartbeat function and solving the problem that SDK will be jammed when plugging out the wire
1.0.10	09-04-2015	Improving IP modification function to solve the problem that modifying IP address is occasionally failed
1.0.12	13-10-2016	Fixing Bugs,added search Ethernet online controllers function and software trigger function etc.
1.0.13	03-07-2017	Added read the specified channel's programmable trigger step count and set the specified channel's tigger mode etc.
1.0.14	05-08-2018	Added set output voltage mode,read output voltage mode,set time unit mode,read time unit mode;Fixed bugs in read maximum current function etc.
1.0.15	15-01-2019	Added USB communication.Added functions OPTController_InitUSB and OPTController_ReleaseUSB.

1 Overview

This programming manual is a specification for OPT Digital Light Source Controller (OPT-DCA24E), which can support both serial port and Ethernet communications (the later is recommended).

1.1 Configuration

The controller has an default static IP address: 192.168.1.16, which can be dynamically allocated by a router. In the case that the IP address of your device(s) is not in the form of 192.168.1.X (X can be any integer within [0, 255]), say 192.168.24.X, we should configure the IP address of the controller accordingly (e.g., 192.168.24.X1). For a switch without DHCP Server (which means it cannot dynamically allocate IP address), we have integrated a tool in our demonstration programm.

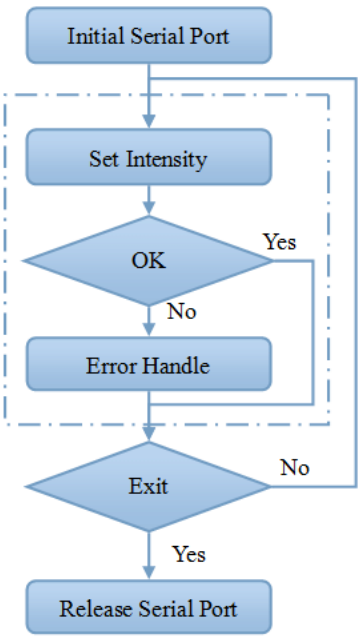
Please note the following things when using the controller:

1. Only one controller can be connected in one time.
2. Make sure that there is no IP address conflict, i.e., one device one IP (including the controller). Otherwise, the connection will not be established.
3. So far, the controller doesn't support wireless connection.

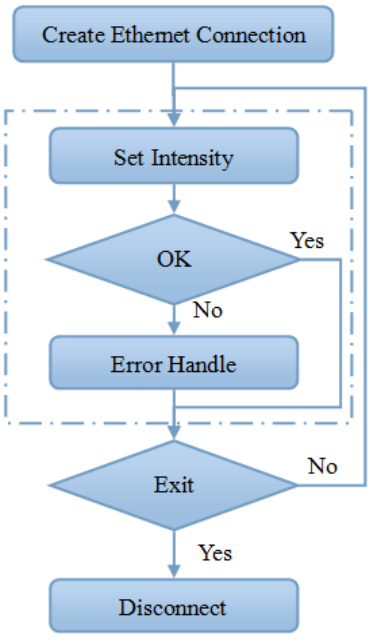
1.2 Controller default settings

1. Baud rate: 9600.
2. The IP address of the controller is dynamically allocated.
3. No check word in communication commands.
4. Back up is enabled when power off .
5. Communication response is enabled.

1.3 Programming flowchart



(a) an example flowchart for serial port communication



(b) an example flowchart for Ethernet communication

Figure 1: Flowcharts for the two types communication (Here, we simply take setting the intensity for example.), respectively. All the steps within dashed rectangle, which are achieved with function codes, are replaceable.

1.4 Example programs

We recommend 20ms time interval between a pair of “Set” and “Read” operations, offering room for the controller to react.

1.4.1 An example in C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace OPTController
{
    class Program
    {
        static private int IntensityValue = 0;
        static void Main(string[] args)
        {
            OPTControllerAPI OptController = new OPTControllerAPI();
            do
            {
                // OptController.InitSerialPort("COM1")
                // create an Ethernet connection by IP address, e.g."192.168.1.16"
                if (0 !=OptController.CreateEthernetConnectionByIP("192.168.1.16") )
                {
                    Console.WriteLine("connection failed");
                    break;
                }

                // Set the intensity 0 to all channels
                if (0 != OptController.SetIntensity( 0, 0))
                {
                    Console.WriteLine("Failed to set intensity for all channels");
                    break;
                }

                // Set the intensity 255 to channel 1
                if (0 != OptController.SetIntensity( 0, 255))
                {
                    Console.WriteLine("Failed to set intensity for the 1st channels");
                    break;
                }

                // Read the intensity of the 1st channel
                if (0 != OptController.ReadIntensity( 1, ref(IntensityValue)))
                {
                    Console.WriteLine(IntensityValue);
                    Console.WriteLine("Failed to read intensity of the 1st channel");
                    break;
                }

                // Turn off all channels
                if (0 != OptController.TurnOffChannel( 0))
                {
                    Console.WriteLine("Failed to turn off all channels");
                    break;
                }
                // Turn on all channels
                if (0 != OptController.TurnOnChannel( 0))
                {
                    Console.WriteLine("Failed to turn on all channels");
                    break;
                }
            } while (false);
            // destroy the connection
            int ret = 0;
            ret = OptController.DestroyEthernetConnect();
            if (0 != ret)
            {
                Console.WriteLine("Failed to destroy the connection");
            }
            else
            {
                Console.WriteLine("DONE");
```

```
69                 }
                   Console.ReadKey();
71             }
           }
73 }
```

Note: Please tick “Allow unsafe code” in (ProjectName — Properties — Build).

1.4.2 An example in VC++

```
1 // connect to controller
  OPTController_InitSerialPort(W2A(strCOMName.GetBuffer(0)), &m_OPTControllerHanlde);
3 // OPTController_CreateEthernetConnectionBySN(W2A(strSN.GetBuffer(0)), &m_OPTControllerHanlde
      );
  // OPTController_DestroyEthernetConnection(m_OPTControllerHanlde);
5
  // turn on the 1st channel
7 OPTController_TurnOnChannel(m_OPTControllerHanlde, 1);
  // turn off the 1st channel
9 OPTController_TurnOffChannel(m_OPTControllerHanlde, 1);

11 // Set the intensity 255 to the 3rd channel
  OPTController_SetIntensity(m_OPTControllerHanlde, 3, 255);
13
  // destroy the connection with the controller
15 // OPTController_DestroyEthernetConnection(m_OPTControllerHanlde);
  OPTController_ReleaseSerialPort(m_OPTControllerHanlde);
```

1.4.3 An example in VB

```
  'Create a connection to the controller
2 Dim IPAddress As String
  IPAddress = "192.168.18.20"
4 Dim controllerHandle As Integer
  OPTControllerAPI.OPTController_CreateEthernetConnectionByIP(IPAddress,controllerHandle)
6
  'Turn on/off  the 1st channel
8 OPTControllerAPI.OPTController_TurnOnChannel(controllerHandle,1)
  OPTControllerAPI.OPTController_TurnOffChannel(controllerHandle,1)
10
  'Set intensity 255 to the 1st channel
12 OPTControllerAPI.OPTController_SetIntensity(controllerHandle,1,255)

14 'Read the intensity of the 1st channel(channel range 1 to 16). Before you read the intensity,
       you need to delay
  Dim nIntensity As Integer
16 Threading.Thread.Sleep(100)
  OPTControllerAPI.OPTController_ReadIntensity(controllerHandle, 1, nIntensity)
18
  'Disconnect the controller
20 OPTControllerAPI.OPTController_DestroyEthernetConnection(controllerHandle)
```

2 Function Specification

2.1 Initializing a serial port(Support OPT controller, SCI visual controller (Q2/Q3/X3))

1. Function: long OPTController_InitSerialPort(char* comName, OPTController_Handle *controllerHandle)
2. Description: initialize an available serial port.
3. Input(s): comName – the name of the serial port. e.g., COM1.
4. Output(s): controllerHandle – a handle of the controller.

5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_INITSERIAL_FAILED or OPT_ERR_SERIALPORT_UNOPENED (see the error code in Tab. 1).
6. See also: releasing a serial port.

2.2 Releasing a serial port(Support OPT controller, SCI visual controller (Q2/Q3/X3))

1. Function: long OPTController_ReleaseSerialPort(OPTController_Handle controllerHandle)
2. Description: release an existing serial port.
3. Input(s)：controllerHandle – the handle of the controller.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_RELEASESERIALPORT_FAILED (see the error code in Tab. 1).
5. See also: initializing a serial port.

2.3 Creating an Ethernet connection (by IP,support OPT controller)

1. Function: long OPTController_CreateEthernetConnectionByIP(char *serverIPAddress, OPTController_Handle *controllerHandle)
2. Description: create an Ethernet connection by IP address.
3. Input(s)：serverIPAddress – the IP of the server. e.g., IP address of the device which is employed as server，The server IP address can be 192.168.1.16.
4. Output(s): controllerHandle – the handle of the controller.
5. Return value：
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_CREATEETHECON_FAILED (see the error code in Tab. 1).
6. Remarks: Connect the controller as a client to the controlled light source device. Before connecting, make sure that the controller is connected to the LAN.
7. See also: destroying an Ethernet connection.

2.4 Creating an Ethernet connection (by SN,support OPT controller)

1. Function: long OPTController_CreateEthernetConnectionBySN(char *serialNumber, OPTController_Handle *controllerHandle)
2. Description: create an Ethernet connection by serial number.
3. Input(s)：serialNumber – the serial number of the controller,such as "AA53017016".
4. Output(s): controllerHandle – the handle of the controller.
5. Return value：
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_CREATEETHECON_FAILED (see the error code in Tab. 1).
6. Remarks:
   - Connect the controller as a client to the controlled light source device. Before connecting, make sure that the controller is connected to the LAN;
   - We recommend creating an Ethernet connection by SN (compared with by IP) because IP is likely to be changed dynamically in LAN under the DHCP protocol. We have provided a tool (SearchForControllers.exe) to check SN.
7. See also: destroying an Ethernet connection.

2.5 Destroying an Ethernet connection(Support OPT controller)

1. Function: long OPTController_DestroyEthernetConnection(OPTController_Handle controllerHandle)
2. Description: disconnect an existing Ethernet Connection.
3. Input(s)：controllerHandle – the handle of the controller.
4. Return value：
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_DESTROYETHECON_FAILED (see the error code in Tab. 1).
5. See also: creating an Ethernet connection.

2.6 Creating an USB connection(Support SCI visual controller (Q2/Q3/X3))

1. Function: long OPTController_InitUSB(char *IDs, OPTController_Handle *controllerHandle)
2. Description: create an USB connection.
3. Input：IDs: – the vendor ID and product ID of the USB device(e.g."1155,22352" –the default vendor ID and product ID of SCI Vision Controller).
4. Output:controllerHandle – the handle of the controller.
5. Return value：
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_INITUSB_FAILED (see the error code in Tab. 1).
6. See also: Release USB connection.

2.7 Releasing USB connection(Support SCI visual controller (Q2/Q3/X3))

1. Function: long OPTController_ReleaseUSB(OPTController_Handle controllerHandle)
2. Description: release USB connection.
3. Input：controllerHandle – the handle of the controller.
4. Return value：
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_RELEASEUSB_FAILED (see the error code in Tab. 1).
5. See also: Creating an USB connection.

2.8 Turning on channel(s)

1. Function: long OPTController_TurnOnChannel(OPTController_Handle controllerHandle, int channelIndex)
2. Description: turn on the specified channel(s).
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s) to be turned on, range: [0 – 16] (in decimal form, 0 for all channels,1-16 represents the index of the specified channel).
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_TURNONCH_FAILED or OPT_ERR_CHINDEX_OUTRANGE (see the error code Tab. 1).
5. See also: turning off channel(s).

2.9 Turning on multiple channels

1. Function: long OPTController_TurnOnMultiChannel(OPTController_Handle controllerHandle, int* channelIndexArray, int length)
2. Description: turn on the specified multiple channels.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndexArray -an array consists of the indexes of the channels to be turned on, range: [1 – 16] (in decimal form,[1 - 16] represents the channel number of the corresponding channel);
   - length – the length of the channel index array.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_TURNONCH_FAILED or OPT_ERR_CHINDEX_OUTRANGE (see the error code in Tab. 1).
5. See also: turning off multiple channels.

2.10 Turning off channel(s)

1. Function: long OPTController_TurnOffChannel(OPTController_Handle controllerHandle, int channelIndex)
2. Description: turn off the specified channel(s).
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s) to be turned off, range: [0 – 16] (in decimal form, 0 for all channels,1-16 represents the index of the specified channel).
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_TURNOFFCH_FAILED or OPT_ERR_CHINDEX_OUTRANGE (see the error code in Tab. 1).
5. See also: turning on channel(s).

2.11 Turning off multiple channels

1. Function: long OPTController_TurnOffMultiChannel(OPTController_Handle controllerHandle, int* channelIndexArray, int length)
2. Description: turn off the specified multiple channels.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndexArray – an array consists of the indexes of the channels to be turned off, range: [1 – 16] (in decimal form,[1 - 16] represents the channel number of the corresponding channel);
   - length – the length of the channel index array.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_TURNOFFCH_FAILED or OPT_ERR_CHINDEX_OUTRANGE (see the error code in Tab. 1).
5. See also: turning on multiple channels.

2.12 Setting intensity

1. Function: long OPTController_SetIntensity(OPTController_Handle controllerHandle, int channelIndex, int intensity)
2. Description: set intensity for the specified channel(s).
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [0 – 16] (in decimal form, 0 for all channels,1-16 for the index of the specified channel);
   - intensity –the intensity value, range: [0 – 255] (in decimal form).
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_INTENSITY_FAILED, OPT_ERR_CHINDEX_OUTRANGE, or OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).
5. See also: reading intensity.

2.13 Setting multiple intensity

1. Function: long OPTController_SetMultiIntensity (OPTController_Handle controllerHandle, IntensityItem* intensityArray, int length)
2. Description: set intensities for the specified one channel or multiple channels.
3. Input(s):
   - controllerHandle – the handle of controller;
   - intensityArray – an array consists of the intensities (and the indexes of the corresponding channels) to be set, range: [0 – 255] (in decimal form);
   - length – the length of the intensity array.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_INTENSITY_FAILED or OPT_ERR_PARAM_OUTRANGE (see the error code Tab. 1).
5. See also: reading intensity.

2.14 Reading intensity

1. Function: long OPTController_ReadIntensity(OPTController_Handle controllerHandle, int channelIndex, int *intensity)
2. Description: read intensity of the specified one channel or channel.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range: [1 – 16] (in decimal form,[1 - 16] represents the channel number of the corresponding channel).
4. Output(s): intensity – the obtained intensity value.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_INTENSITY_FAILED or OPT_ERR_CHINDEX_OUTRANGE (see the error code in Tab. 1).
6. See also: setting intensity.

2.15 Setting the trigger pulse width

1. Function: long OPTController_SetTriggerWidth(OPTController_Handle controllerHandle, int channelIndex, int triggerWidth)
2. Description: set trigger pulse width for corresponding channel(s).
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [0 – 16] (in decimal form, 0 for all channels,1-16 for the index of the specified channel);
   - triggerWidth – the value of the trigger pulse width to be set, range: [1 – 1023],Unit:1ms,Please refer to the specification for details.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_TRIGGERWIDTH_FAILED, OPT_ERR_CHINDEX_OUTRANGE, or OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).
5. See also: reading the trigger pulse width.

2.16 Setting multiple trigger pulse width

1. Function: long OPTController_SetMultiTriggerWidth(OPTController_Handle controllerHandle, TriggerWidthItem* triggerWidthArray, int length)
2. Description: set trigger pulse width for specified the multiple channels.
3. Input(s):
   - controllerHandle – the handle of controller;
   - triggerWidthArray – an array consists of values of the trigger pulse width (and the indexes of the corresponding channels) to be set, range: [1 – 1023],Unit:1ms,Please refer to the specification for details;
   - length – the length of the trigger width array.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_TRIGGERWIDTH_FAILED, OPT_ERR_CHINDEX_OUTRANGE, or OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).
5. See also: reading the trigger pulse width.

2.17 Reading the trigger pulse width

1. Function: long OPTController_ReadTriggerWidth(OPTController_Handle controllerHandle, int channelIndex, int* triggerWidth)
2. Description: read the trigger pulse width of the specified channel
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range: [1 – 16] (in decimal form,[1 - 16] represents the channel number of the corresponding channel).
4. Output(s): triggerWidth –the obtained trigger pulse width.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_TRIGGERWIDTH_FAILED or OPT_ERR_CHINDEX_OUTRANGE (see the error code in Tab. 1).
6. See also: setting the trigger pulse width and setting multiple trigger pulse width.

2.18 Setting high brightness trigger pulse width

1. Function: long OPTController_SetHBTriggerWidth(OPTController_Handle controllerHandle, int channelIndex, int HBTriggerWidth)
2. Description: set high brightness trigger pulse width for corresponding channel(s).
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [0 – 16] (in decimal form, 0 for all channels,1-16 for the index of the specified channel);
   - HBTriggerWidth – the value of the high brightness trigger pulse width to be set, range: [1 – 500],Unit:0.01ms.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_HBTRIGGERWIDTH_FAILED, OPT_ERR_CHINDEX_OUTRANGE, or OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).
5. See also: reading the high brightness trigger pulse width.

2.19 Setting multiple high brightness trigger pulse width

1. Function: long OPTController_SetMultiHBTriggerWidth(OPTController_Handle controllerHandle, HBTriggerWidthItem* HBtriggerWidthArray, int length)
2. Description: set high brightness trigger pulse width for the specified multiple channels.
3. Input(s):
   - controllerHandle – the handle of controller;
   - HBtriggerWidthArray – an array consists of values of the high brightness trigger pulse width (and the indexes of the corresponding channels) to be set, range: [1 – 500],Unit:0.01ms;
   - length – the length of the high brightness trigger width array.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_HBTRIGGERWIDTH_FAILED, OPT_ERR_CHINDEX_OUTRANGE, or OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).
5. See also: reading the high brightness trigger pulse width.

2.20 Reading the high brightness trigger pulse width

1. Function: long OPTController_ReadHBTriggerWidth(OPTController_Handle controllerHandle, int channelIndex, int* HBTriggerWidth)
2. Description: read the high brightness trigger pulse width of the specified channel.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range: [1 – 16] (in decimal form,[1 - 16] represents the channel number of the corresponding channel).
4. Output(s): HBTriggerWidth – the obtained high brightness trigger pulse width.
5. Return value：
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_HBTRIGGERWIDTH_FAILED, OPT_ERR_CHINDEX_OUTRANGE(see the error code in Tab. 1).
6. See also: setting high brightness trigger pulse width and setting multiple high brightness trigger pulse width.

2.21 Enable response

1. Function: OPTController_EnableResponse(OPTController_Handle controllerHandle, bool isResponse)
2. Description: to set whether return value are needed or not.
3. Input:
   - controllerHandle –the handle of controller;
   - isResponse –"true" means "need return value" while "false" stands for not.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_UNKOWN (see the error code in Tab. 1).

2.22 Enable checksum

1. Function: OPTController_EnableCheckSum(OPTController_Handle controllerHandle, bool isCheckSum)
2. Description: to set whether checksum are needed or not.
3. Input:
   - controllerHandle –the handle of controller;
   - isCheckSum –"true" means "need checksum" while "false" stands for not.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_UNKOWN (see the error code in Tab. 1).

2.23 Enable back up when power off

1. Function: OPTController_EnablePowerOffBackup(OPTController_Handle controllerHandle, bool isSave)
2. Description: to set whether backup are needed or not in the case of power off.
3. Input:
   - controllerHandle –the handle of controller;
   - isSave –"true" means "need backup" while "false" stands for not.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_UNKOWN (see the error code in Tab. 1).

2.24 Reading serial number

1. Function: long OPTController_ReadSN(OPTController_Handle controllerHandle, char *SN)
2. Description: read the serial number (SN) of the controller.
3. Input(s): controllerHandle – the handle of controller.
4. Output(s): SN – the obtained serial number.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_SN_FAILED (see the error code in Tab. 1).

2.25 Setting IP configuration

1. Function:long OPTController_SetIPConfiguration(OPTController_Handle controllerHandle, char *IP, char *subnetMask, char *defaultGateway)
2. Description:Set the IP configuration.
3. Input(s):
   - controllerHandle – the handle of controller;
   - IP – Configure the network port IP address;
   - subnetMask – Configure the interface subnet mask;
   - defaultGateway – Configure the network port default gateway.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_IPCONFIG_FAILED (see the error code in Tab. 1).

2.26 Reading IP configuration

1. Function: long OPTController_ReadIPConfig(OPTController_Handle controllerHandle, char *IP, char *subnetMask, char *defaultGateway)
2. Description: read IP configuration of the controller.
3. Input(s): controllerHandle – the handle of controller.
4. Output(s):
   - IP – the obtained IP address;
   - subnetMask – the obtained subnet mask;
   - defaultGateway – the obtained default gateway.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_IPCONFIG_FAILED (see the error code in Tab. 1).

2.27 Setting maximum current

1. Function: long OPTController_SetMaxCurrent (OPTController_Handle controllerHandle, int channelIndex, int current)
2. Description: set maximum current for the specified channel(s).
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [0 – 16] (in decimal form, 0 for all channels,1-16 for the index of the specified channel);
   - current – the value of the maximum current to be set, range: [1 – 200] (in decimal form),Unit:10mA.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_MAXCURRENT_FAILED (see the error code in Tab. 1).
5. See also: reading maximum current.

2.28 Setting multiple maximum current

1. Function: OPTController_SetMultiMaxCurrent(OPTController_Handle controllerHandle, MaxCurrentItem *maxCurrentArray, int length);
2. Description: set maximum current for the specified multiple channels.
3. Input(s):
   - controllerHandle – the handle of controller;
   - maxCurrentArray – an array consists of values of the maximum current (and the indexes of corresponding channels) to be set, range: [1 – 200],Unit:10mA;
   - length – the length of the maximum current array.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_MAXCURRENT_FAILED (see the error code in Tab. 1).
5. See also: reading maximum current.

2.29 Reading maximum current

1. Function: OPTController_ReadMaxCurrent (OPTController_Handle controllerHandle, int channelIndex, int mode, int *value)
2. Description: read maximum current for the specified channel.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range: [1 – 16] (in decimal form,[1 - 16] represents the channel number of the corresponding channel).
   - mode – the mode of value,range:[0 -2], 0:Read manually set current value;1:Read the current current value;2:Read voltage value.
4. Output(s): value – the obtained value.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_MAXCURRENT_FAILED (see the error code in Tab. 1).
6. See also: setting maximum current and setting multiple maximum current.

2.30 Setting Output voltage

1. Function: OPTController_SetOutPutVoltage(OPTController_Handle controllerHandle, int channelIndex,int voltage)
2. Description: Set the specified channel output voltage.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range:[0-4] (in decimal form,0 for all channels);
   - voltage – the value of voltage.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_VOLTAGE_FAILED (see the error code in Tab. 1).
5. See also: Reading output voltage.

2.31 Reading output voltage

1. Function: OPTController_ReadOutputVoltage(OPTController_Handle controllerHandle, int channelIndex,int *voltage)
2. Description: Read the specified channel output voltage.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range:[1-4] (in decimal form).
4. Output(s): voltage – the voltage value of specified channel.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_VOLTAGE_FAILED (see the error code in Tab. 1).
6. See also: Setting output voltage.

2.32 Reading MAC

1. Function: OPTController_ReadMAC(OPTController_Handle controllerHandle, char *MAC)
2. Description: read the media access control (MAC) address of the controller.
3. Input(s): controllerHandle – the handle of controller.
4. Output: MAC – the obtained media access control address.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_MAC_FAILED (see the error code in Tab. 1).

2.33 Setting trigger activation

1. Function: OPTController_SetTriggerActivation(OPTController_Handle controllerHandle, int channelIndex, int triggerActivation)
2. Description: set the trigger activation of the controller.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range: [0 – 16] (in decimal form,0 for all channels,[1 - 16] represents the channel number of the corresponding channel);
   - triggerActivation – the value of the trigger activation to be set. range: [0 – 3](0:Switched mode(positive);1:Switched mode(negative);2:Pulsed mode(Falling Edge); 3:Pulsed mode(Rising Edge)).Network port controller only supports two trigger modes,rising edge trigger and falling falling edge trigger,and the default is rising edge trigger.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_TRIGGERACTIVATION_FAILED (see the error code in Tab. 1).
5. See also: reading trigger activation.

2.34 Reading trigger activation

1. Function: OPTController_ReadTriggerActivation(OPTController_Handle controllerHandle, int channelIndex, int *triggerActivation)
2. Description: read the trigger activation of the controller.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index of the channel, range: [1 – 16] (in decimal form,,[1 - 16] represents the channel number of the corresponding channel).
4. Output: triggerActivation – the obtained trigger activation.range: [0 – 3](0:Switched mode(positive);1:Switched mode(negative);2:Pulsed mode(Falling Edge); 3:Pulsed mode(Rising Edge));Network port controller only supports two trigger modes,rising edge trigger and falling falling edge trigger,and the default is rising edge trigger.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_TRIGGERACTIVATION_FAILED (see the error code in Tab. 1).
6. See also: setting trigger activation.

2.35 Setting work mode

1. Function: OPTController_SetWorkMode(OPTController_Handle controllerHandle, int workMode);
2. Description: set the work mode of the controller.
3. Input(s):
   - controllerHandle – the handle of controller;
   - workMode – the value of the work mode to be set, range: [0 – 3](0 for General Lighting Mode,1 for General Trigger Mode,2 for Highlight Trigger Mode,3 for Set the working mode on pannel).
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_WORKMODE_FAILED (see the error code in Tab. 1).
5. See also: reading work mode.

2.36 Reading work mode

1. Function: OPTController_ReadWorkMode(OPTController_Handle controllerHandle, int *workMode);
2. Description: read the work mode of the controller.
3. Input(s): controllerHandle – the handle of controller.
4. Output: workMode – the obtained work mode.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_WORKMODE_FAILED (see the error code in Tab. 1).
6. See also: setting work mode.

2.37 Setting outer trigger frequency upper bound

1. Function: OPTController_SetOuterTriggerFrequencyUpperBound(OPTController_Handle controllerHandle, int channelIndex, int maxFrequency);
2. Description: set the outer trigger frequency upper bound of the controller.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [0 – 16] (in decimal form, 0 for all channels,1-16 for the index of the specified channel);
   - maxFrequency – the obtained maximum frequency,range:[1–900],Unit:1HZ.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_OUTERTRIGGERFREQUENCYUPPERBOUND_FAILED (see the error code in Tab. 1).

2.38 Reading outer trigger frequency upper bound

1. Function: OPTController_ReadOuterTriggerFrequencyUpperBound(OPTController_Handle controllerHandle, int channelIndex, int *maxFrequency);
2. Description: read the outer trigger frequency upper bound of the controller.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [1 – 16] (in decimal form, 1-16 for the index of the specified channel);
4. Output: maxFrequency – the obtained maximum frequency.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_OUTERTRIGGERFREQUENCYUPPERBOUND_FAILED (see the error code in Tab. 1).

2.39 Automatically detecting the load once

1. Function: OPTController_AutoDetectLoadOnce(OPTController_Handle controllerHandle);
2. Description: automatically detect the load of the controller once.
3. Input(s): controllerHandle – the handle of controller.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_WRITE_FAILED (see the error code in Tab. 1).

2.40 Setting auto strobe frequency

1. Function: OPTController_SetAutoStrobeFrequency(OPTController_Handle controllerHandle, int channelIndex, int frequency);
2. Description: set the auto-strobe frequency of the controller,only valid for stroboscopic controllers.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [0 – 16] (in decimal form, 0 for all channels,1-16 for the index of the specified channel);

- frequency – the value of the frequency to be set, range: [15 – 1000],Unit:1Hz.

4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_AUTOSTROBEFREQUENCY_FAILED (see the error code in Tab. 1).

2.41 Reading auto strobe frequency

1. Function: OPTController_ReadAutoStrobeFrequency(OPTController_Handle controllerHandle, int channelIndex, int *frequency);
2. Description: Read the auto-strobe frequency of the controller.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [1 – 16] (in decimal form, 1-16 for the index of the specified channel).
4. Output(s): frequency – get the value of the frequency.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_AUTOSTROBEFREQUENCY_FAILED (see the error code in Tab. 1).

2.42 Enable DHCP

1. Function: OPTController_EnableDHCP(OPTController_Handle controllerHandle, BOOL bDHCP);
2. Description: to enable or disable the Dynamic Host Configuration Protocol (DHCP).
3. Input(s):
   - controllerHandle – the handle of controller;
   - bDHCP – "TRUE" means "enable DHCP" while "FALSE" stands for disable.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_DHCP_FAILED (see the error code in Tab. 1).

2.43 Setting load mode

1. Function: OPTController_SetLoadMode(OPTController_Handle controllerHandle,int channelIndex, int loadMode);
2. Description: set the load mode of the controller
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range: [0 – 16] (in decimal form, 0 for all channels,1-16 for the index of the specified channel);
   - loadMode – the value of the load mode to be set, 0 for setting detect automatically, 1 for setting manually.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_LOADMODE_FAILED (see the error code in Tab. 1).

2.44 Reading properties

1. Function: OPTController_ReadProperties(OPTController_Handle controllerHandle, int property, char *value);
2. Description: read the property of the controller.
3. Input(s):
   - controllerHandle – the handle of controller;
   - property – the code of the property to be read,1 for controller model, 2 for controller firmware version.
4. Output(s): value – the obtained property.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READ_PROPERTY_FAILED (see the error code in Tab. 1).

2.45 Getting vision of controller DLL

1. Function: OPTController_GetVersion(char *version);
2. Description: get version of the controller DLL.
3. Output(s): version – the version of controller DLL.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_UNKNOWN.

2.46 Resetting connection by SN

1. Function: long OPTController_ConnectionResetBySN(char *serialNumber);
2. Description: reset the connection of the controller with the specified serial number using User Datagram Protocol(UDP).
3. Input(s): serialNumber– the serial number of the connection.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_CONNECTION_RESET_FAILED (see the error code in Tab. 1).
5. Remarks: For current controller, this operation will cost 150ms.
6. See also: Resetting connection by IP address.

2.47 Resetting connection by IP address

1. Function: long OPTController_ConnectionResetByIP(char *serverIPAddress)
2. Description: reset the connection of the controller with the specified IP address using User Datagram Protocol(UDP).
3. Input(s): serverIPAddress– the controller uses the IP address to reset connection.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_CONNECTION_RESET_FAILED (see the error code in Tab. 1).
5. See also: Resetting connection by SN.

2.48 Setting heartbeat function

1. Function: long OPTController_SetEthernetConnectionHeartBeat(OPTController_Handle controllerHandle, unsigned timeout);
2. Description: Setting heartbeat function after establishing connection;Only valid for the network port controller,and the SDK default setting time of heartbeat package is 5s.
3. Input(s)：
   - controllerHandle – the handle of controller;
   - timeout – heartbeat timeout,range:[1-65535] Unit: 1second(S). if timeout=0, heartbeat packet will not be send. When timeout>0, it is recommended to send heartbeat packet every timeout/2 second.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SET_HEARTBEAT_FAILED (see the error code in Tab. 1).

2.49 Checking the controller is connected

1. Function: long OPTController_IsConnect(OPTController_Handle controllerHandle);
2. Description: check the controller connect state.
3. Input(s): controllerHandle – the handle of controller.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_UNKNOWN (see the error code in Tab. 1).

2.50 Getting channel state

1. Function: long OPTController_GetChannelState(OPTController_Handle controllerHandle, int channelIndex, int *state);
2. Description: Get channel state.
3. Input(s)
   - controllerHandle – the handle of controller;
   - channelIndex – index of channel,channel serial number range:[1 - 16](in decimal form,1 - 16 represents the channel serial number of the corresponding channel).
4. Output(s):state – the channel state; 0 –Light sources connected;1 –Light sources disconnected;2 –Short circuit protection;3 – Over voltage protection;4 –Over current protection
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_GET_CHANNELSTATE_FAILED (see the error code in Tab. 1).

2.51 Getting controller SN on ethernet

1. Function: long OPTController_GetControllerListOnEthernet(char *snList);
2. Description: search for the online controllers, get their SNs.
3. Output(s):snList – Get the on-line controller serial numbers, serial numbers with a comma separated，like ”AA53017016,AA54278910”.
4. Return value：
   - succeed:OPT_SUCCEED;
   - failed:OPT_ERR_GETCONTROLLERLIST_FAILED (see the error code in Tab. 1).

2.52 Setting keepalive parameter(controller's firmware version should be V3.2.7 and above)

1. Function:long OPTController_SetKeepaliveParameter(OPTController_Handle controllerHandle, int keepalive_time, int keepalive_intvl, int keepalive_probes);
2. Description:set keepalive parameters.
3. Input(s):
   - controllerHandle –the handle of controller;
   - keepalive_time –idle time,range:[1-65535] (in decimal form),Unit:seconds,default:5s;
   - keepalive_intvl –interval between two keepalive_probes,range:[1-65535] (in decimal form),Unit:seconds,default:3s;
   - keepalive_probes –probes of keepalive range:[1-65535],default:9 times.
4. Return value:
   - succeed:OPT_SUCCEED;
   - failed:OPT_ERR_SET_KEEPALIVEPARAMETERS_FAILED (see the error code in Tab. 1).

2.53 Enable/Disable controller keepalive(controller's firmware version should be V3.2.7 and above)

1. Function： long OPTController_EnableKeepalive(OPTController_Handle controllerHandle, BOOL enable);
2. Description： enable or disable keepalive.
3. Input(s):
   - controllerHandle – the handle of controller;
   - enable – "TRUE" means "enable keepalive" while "FALSE" stands for disable.
4. Return value:
   - succeed:OPT_SUCCEED;
   - failed:OPT_ERR_ENABLE_KEEPALIVE_FAILED (see the error code in Tab. 1).

2.54 Setting software trigger(controller's firmware version should be V3.3.1 and above)

1. Function： long OPTController_SoftwareTrigger(OPTController_Handle controllerHandle, int channelIndex, int time);
2. Description:software trigger,specified channel on specified time.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s), range:[0-16] (in decimal form,0 for all the channels,1-16 for the index of the specified channel);
   - time – light duration,range:[1-3000],Unit:10ms.
4. Return value:
   - succeed:OPT_SUCCEED;
   - failed:OPT_ERR_SOFTWARETRIGGER_FAILED (see the error code in Tab. 1).

2.55 Setting multiple software trigger(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_MultiSoftwareTrigger(OPTController_Handle controllerHandle, SoftwareTriggerItem* softwareTriggerArray, int length);
2. Description: software trigger for the specified multiple channels.
3. Input(s):
   - controllerHandle – the handle of controller;
   - softwareTriggerArray – an array consists of values of the software trigger (and the indexes of corresponding channels) to be set, range:[1-3000],Unit:10ms;
   - length – the length of the softwareTrigger array.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SOFTWARETRIGGER_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).
5. See also: Set software trigger.

2.56 Reading programmable trigger step count(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_ReadStepCount(OPTController_Handle controllerHandle, int moduleIndex, int* count)
2. Description: read the specified module's programmable trigger step count.
3. Input(s):
   - controllerHandle – the handle of controller;
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16).
4. Output:count – the specified channel's stepCount.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READSTEPCOUNT_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.57 Setting programmable trigger mode(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_SetTriggerMode(OPTController_Handle controllerHandle,int moduleIndex, int mode);
2. Description: set the specified module's trigger mode.
3. Input(s):
   - controllerHandle – the handle of controller;
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16);
   - mode – trigger mode,range:[1-2]. 1:General-Trigger-mode; 2:Programmable-Trigger-mode.
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SETTRIGGERMODE_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.58 Reading programmable trigger mode(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_ReadTriggerMode(OPTController_Handle controllerHandle, int moduleIndex, int *mode);
2. Description: read the specified module's trigger mode.
3. Input(s):
   - controllerHandle – the handle of controller;
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16).
4. Output: mode – trigger mode,range:[1-2]. 1:General-Trigger-mode; 2:Programmable-Trigger-mode.
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READTRIGGERMODE_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.59 Setting programmable trigger current step index(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_SetCurrentStepIndex(OPTController_Handle controllerHandle, int moduleIndex, int curStepIndex);
2. Description: set the specified module's current step index.
3. Input(s):
   - controllerHandle – the handle of controller;
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16);
   - curStepIndex – the specified channel's current step index,range:[1– 64].
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SETCURRENTSTEPINDEX_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.60 Reading programmable trigger current step index(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_ReadCurrentStepIndex(OPTController_Handle controllerHandle, int moduleIndex, int* curStepIndex);
2. Description: read the specified module's current step index.
3. Input(s):
   - controllerHandle – the handle of controller;
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16).
4. Output: curStepIndex – the specified channel's current step index,range:[1– 64].
5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READCURRENTSTEPINDEX_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR
     OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.61 Reseting programmable trigger current module index(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_ResetSEQ(OPTController_Handle controllerHandle, int moduleIndex);
2. Description: reset the specified module's SEQ.
3. Input(s):
   - controllerHandle – the handle of controller;
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16).
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_RESETSEQ_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.62 Setting SEQ table(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_SetSeqTable(OPTController_Handle controllerHandle,int seqCount,int moduleIndex,int * triggerSource,int *intensity,int *pulseWidth);
2. Description: set the specified module's SEQ table data.
3. Input(s):
   - controllerHandle – the handle of controller;
   - seqCount – the number of seq, range: [1 – 64] (in decimal form);
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16);
   - triggerSource – the trigger source data,range[1-16](the value of moduleIndex is 1,the range of tiggerSource is[1-4];the value of moduleIndex is 2,the range of triggerSource is[5-8];the value of moduleIndex is 3,the range of triggerSource is[9-12];the value of moduleIndex is 4,the range of triggerSource is[13-16]);
   - intensity – the intensity data,range:[0 – 255];
   - pulseWidth – the pulse width data,range:[0 – 1023].
4. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_SETSEQTABLEDATA_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.63 Reading SEQ table(controller's firmware version should be V3.3.1 and above)

1. Function: long OPTController_ReadSeqTable(OPTController_Handle controllerHandle, int moduleIndex,int *seqCount,char *seqTableData);
2. Description: Read the specified module's SEQ table data.
3. Input(s):
   - controllerHandle – the handle of controller;
   - moduleIndex – the index(es) of the module(s), range: [1 – 4] (in decimal form,module 1 includes 1-4 channels,module 2 includes 5-8 channels,module 3 includes 9-12,module 4 includes 13-16).
4. Output(s)
   - seqCount – the number of seq, range: [1 – 64] (in decimal form);

- triggerSource – the trigger source data,range[1-16](the value of moduleIndex is 1,the range of tiggerSource is[1-4];the value of moduleIndex is 2,the range of triggerSource is[5-8];the value of moduleIndex is 3,the range of triggerSource is[9-12];the value of moduleIndex is 4,the range of triggerSource is[13-16]);
- intensity – the intensity data,range:[0 – 255];
- pulseWidth – the pulse width data,range:[0 – 1023].

5. Return value:
   - succeed: OPT_SUCCEED;
   - failed: OPT_ERR_READSEQTABLEDATA_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.64 Setting trigger delay(controller's firmware version should be V3.3.1 and above)

1. Function：long OPTController_SetTriggerDelay(OPTController_Handle controllerHandle, int channelIndex,int triggerDelay);
2. Description：Set the specified channel's trigger delay.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s),range:[0 – 16] (in decimal form,0 for all channels,1-16 for the index of the specified channel);
   - triggerDelay – the trigger delay,if it less than 10,set it 0， range:[0 – 65000],Unit:1us.
4. Return Value:
   - succeed: OPT_SUCCEED;
   - failed：OPT_ERR_SETTRIGGERDELAY_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.65 Getting trigger delay(controller's firmware version should be V3.3.1 and above)

1. Function：long OPTController_GetTriggerDelay(OPTController_Handle controllerHandle, int channelIndex,int* triggerDelay);
2. Description：get the specified channel's trigger delay.
3. Input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s),range:[1 – 16] (in decimal form,1-16 for the index of the specified channel).
4. Output(s)：triggerDelay – the trigger delay,range:[0-65000]， Unit:1us.
5. Return Value:
   - succeed: OPT_SUCCEED;
   - failed：OPT_ERR_GETTRIGGERDELAY_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).

2.66 Setting multiple trigger delay(controller's firmware version should be V3.3.1 and above)

1. Function：long OPTController_SetMultiTriggerDelay(OPTController_Handle controllerHandle, TriggerDelayItem* triggerDelayArray, int length);
2. Description：set multiple channels' trigger delay at a same time.
3. Input(s)：
   - controllerHandle – the handle of controller;
   - triggerDelayArray – an array consists of values of the trigger delay(and the indexes of corresponding channels)to be set,range:[0 – 65000] (in decimal form)， Unit：1us,if the trigger delay less than 10,set it 0;
   - length – the length of the trigger delay array.
4. Return Value：
   - succed：OPT_SUCCEED;
   - failed：OPT_ERR_SOFTWARETRIGGER_FAILED, OPT_ERR_CHINDEX_OUTRANGE, OR OPT_ERR_PARAM_OUTRANGE (see the error code in Tab. 1).
5. See also：Set trigger delay.

2.67 Getting channels of controller

1. Function：long OPTController_GetControllerChannels(OPTController_Handle controllerHandle, int *channels);
2. Description：get the channels of controller.
3. Input(s)：controllerHandle – the handle of controller.
4. Output(s)：channels –the channels of controller.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_CHANNELS_FAILED (see the error code in Tab. 1).

2.68 Reading switch state of keepalive

1. Function：long OPTController_ReadKeepaliveSwitchState(OPTController_Handle controllerHandle, int *state);
2. Description：Read the switch state of keepalive.
3. Input(s)：controllerHandle – the handle of controller.
4. Output(s)：state – the switch state of keepalive.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_KEEPALIVE_STATE_FAILED (see the error code in Tab. 1).

2.69 Reading continuous keepalive time

1. Function：long OPTController_ReadContinuousKeepaliveTime(OPTController_Handle controllerHandle, int *time);
2. Description：read the continuous time of keepalive.
3. Input(s)：controllerHandle – the handle of controller.
4. Output(s)：time – the continuous keepalive time.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_KEEPALIVE_CONTINUOUS_TIME_FAILED (see the error code in Tab. 1).

2.70 Reading delivery times of prop packet

1. Function：long OPTController_ReadPacketDeliveryTimes(OPTController_Handle controllerHandle, int *times);
2. Description：read delivery times of prop packet.
3. Input(s)：controllerHandle – the handle of controller.
4. Output：times – the delivery times of prop packet.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_DELIVERY_TIMES_FAILED (see the error code in Tab. 1).

2.71 Reading interval time of prop packet deliveried

1. Function：long OPTController_ReadIntervalTimeOfPropPacket(OPTController_Handle controllerHandle, int *time);
2. Description：read the interval time of prop packet deliveried.
3. Input(s)：controllerHandle – the handle of controller.
4. Output(s)：times – the interval time of delivery prop packet.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_INTERVAL_TIME_FAILED (see the error code in Tab. 1).

2.72 Reading version of output board

1. Function：long OPTController_ReadOutputBoardVersion(OPTController_Handle controllerHandle, char *version)
2. Description：read the version of output board.
3. Input(s)：controllerHandle – the handle of controller.
4. Output(s)：version – the version of output board.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_OUTPUTBOARD_VISION_FAILED (see the error code in Tab. 1).

2.73 Reading load detection mode

1. Function：long OPTController_ReadLoadDetectMode(OPTController_Handle controllerHandle, int channelIndex,int *mode);
2. Description：read the load detection mode.
3. Input(s)：
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s),range:[1 – 16] (in decimal form,1-16 for the index of the specified channel).
4. Output(s)：mode – load mode，0 for automatic detection，1 for manually setting maximum current.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_DETECT_MODE_FAILED (see the error code in Tab. 1).

2.74 Setting boot state mode

1. Function：long OPTController_SetBootState(OPTController_Handle controllerHandle,int channelIndex, int mode);
2. Description：set the specified channel's open mode boot state.
3. Input(s)：
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s),range:[1 – 16] (in decimal form,1-16 for the index of the specified channel).
   - mode – boot protection mode，0 for general turn on mode，1 for general turn off mode,default set general close mode.
4. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_SET_BOOT_STATE_MODE_FAILED (see the error code in Tab. 1).

2.75 Reading model boot state

1. Function：long OPTController_ReadModelBootState(OPTController_Handle controllerHandle, int channelIndex,int *state);
2. Description：read the specified channel's boot state of model.
3. Input(s)：
   - controllerHandle – the handle of controller；
   - channelIndex – the index(es) of the channel(s),range:[1 – 16] (in decimal form,1-16 for the index of the specified channel).
4. Output(s)：state – boot state，0 for general turn on mode，1 for general turn off mode.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_MODEL_BOOT_MODE_FAILED (see the error code in Tab. 1).

2.76 Setting time unit

1. Function: long OPTController_SetTimeUnit(OPTController_Handle controllerHandle,int channelIndex,int timeUnit);
2. Description：Common trigger mode time unit switching.
3. input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s),range:[1 – 16] (in decimal form,1-16 for the index of the specified channel);
   - timeUnit – the time unit,range:[0 – 3],0 for 1 us,1 for 10 us,2 for 1 ms,3 for 100 ms.
4. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_SET_TIMEUNIT_FAILED (see the error code in Tab. 1).

2.77 Reading time unit

1. Function: long OPTController_ReadTimeUnit(OPTController_Handle controllerHandle,int channelIndex,int *timeUnit);
2. Description：Read the specified channel's common trigger mode time unit.
3. input(s):
   - controllerHandle – the handle of controller;
   - channelIndex – the index(es) of the channel(s),range:[1 – 16] (in decimal form,1-16 for the index of the specified channel).
4. output(s): timeUnit – the time unit,range:[0 – 3],0 for 1 us,1 for 10 us,2 for 1 ms,3 for 100 ms.
5. Return Value：
   - success：OPT_SUCCEED;
   - failed：OPT_ERR_READ_TIMEUNIT_FAILED (see the error code in Tab. 1).

Appendices

A FAQ (frequently asked questions)

A.1 Why the controller dose not respond properly to continuous operations?

We recommend 20ms time interval between a pair of "Set" and "Read" operations, offering room for the controller to react.

A.2 Are there long delays during Ethernet connection?

No, there aren't. If the connection isn't established within 50ms (i.e. the timeout is 50ms.), then it failed.

A.3 Why the functions return error codes while in fact the corresponding operations are successfully done?

Please check whether the responses from the functions are enabled (see how to enable response in Sect. 2.21).

A.4 Why the controller can't find any available PC serial port or serial port connection can't be established?

If the PC is equipped with WIN7 OS, please try to run as administrator.

A.5 Why the DEMO programm can't be opened or the SDK can't be called (system errors are reported.)?

Please install VS2008 runtime library for solution.

A.6 Why the controller can't be properly connected after modifying its MAC address?

Please reboot the controller after modifying its MAC address.

A.7 Why the heart beat function is necessary?

When the Ethernet connection is broken (e.g. due to that the program is closed abnormally or physical cause of internet) but the controller is still live, it can't judge whether the TCP/IP connection is still valid. With the heart beat function, the client will send heart beat packets once in a while (determined by heartbeat timeout) to the controller and the controller will check whether the connection is still existing. In the version after 1.0.8, by default, the heart beat function is enabled (See Sect. 2.48).

A.8 How to send effective heart beat packet?

Generally, threading are used to sent heart beat packet in a certain interval(i.e. half of the heartbeat timeout. E.g., if the heartbeat timeout is 5s, the interval will be 2.5s. The command word for heart beat function is 0x46.). Make sure that the controller can receive the heart beat packet within the interval, otherwise the controller will be disconnected. One negative scenario is when the threading for sending heart beat packet has a low priority and the CPU is occupied by other threading. The heart beat packet can not be sent on time. Similarly, if the priority for the timer threading is low, most likely the same story will happen.

A.9 Why commands are failed occasionally during communication?

The controller needs a response time for each command. If a new command arrives when the controller is processing one command, the controller will give up the old command. Therefore, to ensure the correct responses from the controller the following are suggested:

- Setting delay after time-consuming command;
- Enabling response for each command to check whether the controller has finished processing the current command;
- The communication should be in synchronous blocking mode.

A.10 In Ethernet communication, why wrong controller is connected when there are several controllers present, i.e., the SN (or IP address) of the connected controller is not the same as the expected?

This may due to that several controllers share the same IP address. Notice that each controller has an static IP address, 192.168.1.16, by default. In LAN, please make sure the IP address uniqueness for each controller.

A.11 Why the controller is available yet can't be connected?

Please check whether the client and the controller are in the the same Ethernet segment.

B Macro Definitions for Error Codes

Table 1: Error code

Macro Name	Error Code	Remarks
OPT_SUCCEED	0	Operation succeed
OPT_ERR_INVALIDHANDLE	3001001	Invalid handle
OPT_ERR_UNKNOWN	3001002	Error unknown
OPT_ERR_INITSERIAL_FAILED	3001003	Failed to initialize a serial port
OPT_ERR_RELEASESERIALPORT_FAILED	3001004	Failed to release a serial port
OPT_ERR_SERIALPORT_UNOPENED	3001005	Attempt to access an unopened serial port
OPT_ERR_CREATEETHECON_FAILED	3001006	Failed to create an Ethernet connection
OPT_ERR_DESTROYETHECON_FAILED	3001007	Failed to destroy an Ethernet connection
OPT_ERR_SN_NOTFOUND	3001008	SN is not found
OPT_ERR_TURNONCH_FAILED	3001009	Failed to turn on the specified channel(s)
OPT_ERR_TURNOFFCH_FAILED	3001010	Failed to turn off the specified channel(s)
OPT_ERR_SET_INTENSITY_FAILED	3001011	Failed to set the intensity for the specified channel(s)
OPT_ERR_READ_INTENSITY_FAILED	3001012	Failed to read the intensity for the specified channel
OPT_ERR_SET_TRIGGERWIDTH_FAILED	3001013	Failed to set trigger pulse width
OPT_ERR_READ_TRIGGERWIDTH_FAILED	3001014	Failed to read trigger pulse width
OPT_ERR_SET_HBTRIGGERWIDTH_FAILED	3001015	Failed to set high brightness trigger pulse width
OPT_ERR_READ_HBTRIGGERWIDTH_FAILED	3001016	Failed to read high brightness trigger pulse width
OPT_ERR_READ_SN_FAILED	3001017	Failed to read serial number of the controller
OPT_ERR_READ_IPCONFIG_FAILED	3001018	Failed to read IP configuration of the controller
OPT_ERR_CHINDEX_OUTRANGE	3001019	Index(es) of channel(s) out of the range
OPT_ERR_WRITE_FAILED	3001020	Failed to write data

Table 1 – continued from previous page

Macro Name	Error Code	Remarks
OPT_ERR_PARAM_OUTRANGE	3001021	Parameter(s) out of the range
OPT_ERR_READ_MAC_FAILED	3001022	Failed to read MAC
OPT_ERR_SET_MAXCURRENT_FAILED	3001023	Failed to set max current
OPT_ERR_READ_MAXCURRENT_FAILED	3001024	Failed to read max current
OPT_ERR_SET_TRIGGERACTIVATION_FAILED	3001025	Failed to set trigger activation
OPT_ERR_READ_TRIGGERACTIVATION_FAILED	3001026	Failed to read trigger activation
OPT_ERR_SET_WORKMODE_FAILED	3001027	Failed to set work mode
OPT_ERR_READ_WORKMODE_FAILED	3001028	Failed to read work mode
OPT_ERR_SET_BAUDRATE_FAILED	3001029	Failed to set baud rate
OPT_ERR_SET_CHANNELAMOUNT_FAILED	3001030	Reserved
OPT_ERR_SET_DETECTEDMINLOAD_FAILED	3001031	Reserved
OPT_ERR_READ_OUTERTRIGGERFREQUENCYUPPERBOUND_FAILED	3001032	Failed to read outer trigger frequency upper bound
OPT_ERR_SET_AUTOSTROBEFREQUENCY_FAILED	3001033	Failed to set auto strobe frequency
OPT_ERR_READ_AUTOSTROBEFREQUENCY_FAILED	3001034	Failed to read auto strobe frequency
OPT_ERR_SET_DHCP_FAILED	3001035	Failed to set DHCP
OPT_ERR_SET_LOADMODE_FAILED	3001036	Failed to set load mode
OPT_ERR_READ_PROPERTY_FAILED	3001037	Failed to read property
OPT_ERR_CONNECTION_RESET_FAILED	3001038	Failed to reset connection
OPT_ERR_SET_HEARTBEAT_FAILED	3001039	Failed to set Ethernet connection heartbeat
OPT_ERR_GETCONTROLLERLIST_FAILED	3001040	Failed to get controler(s) list
OPT_ERR_SOFTWARETRIGGER_FAILED	3001041	Failed to software_trigger
OPT_ERR_GET_CHANNELSTATE_FAILED	3001042	Failed to get channelstate
OPT_ERR_SET_KEEPALIVEPARAMETERS_FAILED	3001043	Failed to set keepalvie parameters
OPT_ERR_ENABLE_KEEPALIVE_FAILED	3001044	Failed to enable/disable keepalive
OPT_ERR_READSTEPCOUNT_FAILED	3001045	Failed to read step count
OPT_ERR_SETTRIGGERMODE_FAILED	3001046	Failed to set trigger mode
OPT_ERR_READTRIGGERMODE_FAILED	3001047	Failed to read trigger mode
OPT_ERR_SETCURRENTSTEPINDEX_FAILED	3001048	Failed to set current step index
OPT_ERR_READCURRENTSTEPINDEX_FAILED	3001049	Failed to read current step index
OPT_ERR_RESETSEQ_FAILED	3001050	Failed to reset the specified channel's SEQ
OPT_ERR_SETTRIGGERDELAY_FAILED	3001051	Failed to set trigger delay
OPT_ERR_GETTRIGGERDELAY_FAILED	3001052	Failed to read trigger delay
OPT_ERR_SETMULTITRIGGERDELAY_FAILED	3001053	Failed to set multiple channels trigger delay

Table 1 – continued from previous page

Macro Name	Error Code	Remarks
OPT_ERR_SETSEQTABLEDATA_FAILED	3001054	Failed to set SEQ table data
OPT_ERR_READSEQTABLEDATA_FAILED	3001055	Failed to read SEQ table data
OPT_ERR_READ_CHANNELS_FAILED	3001056	Failed to read controller's channel
OPT_ERR_READ_KEEPALIVE_STATE_FAILED	3001057	Failed to read the state of keepalive
OPT_ERR_READ_KEEPALIVE_CONTINUOUS_TIME_FAILED	3001058	Failed to read the continuous time of keepalive
OPT_ERR_READ_DELIVERY_TIMES_FAILED	3001059	Failed to read the delivery times of prop packet
OPT_ERR_READ_INTERVAL_TIME_FAILED	3001060	Failed to read the interval time of prop packet
OPT_ERR_READ_OUTPUTBOARD_VISION_FAILED	3001061	Failed to read the vision of output board
OPT_ERR_READ_DETECT_MODE_FAILED	3001062	Failed to read detect mode of load
OPT_ERR_SET_BOOT_STATE_MODE_FAILED	3001063	Failed to set mode of boot state
OPT_ERR_READ_MODEL_BOOT_MODE_FAILED	3001064	Failed to read the specified channel boot state
OPT_ERR_SET_OUTERTRIGGERFREQUENCYUPPERBOUND_FAILED	3001065	Failed to set outer trigger frequency upper bound
OPT_ERR_SET_IPCONFIG_FAILED	3001066	Failed to set up the network port configuration
OPT_ERR_SET_VOLTAGE_FAILED	3001067	Failed to set voltage value
OPT_ERR_READ_VOLTAGE_FAILED	3001068	Failed to read voltage value
OPT_ERR_SET_TIMEUNIT_FAILED	3001069	Failed to set time unit
OPT_ERR_READ_TIMEUNIT_FAILED	3001070	Failed to read time unit
OPT_ERR_CHANNEL_SHORT_FAILED	3001071	Short circuit protection
OPT_ERR_CHANNEL_NULL_FAILED	3001072	No light connected
OPT_ERR_INITUSB_FAILED	3001073	Failed to initialize an USB connection
OPT_ERR_RELEASEUSB_FAILED	3001074	Failed to release USB
OPT_ERR_USBREADTRIGGERWIDTH_FAILED	3001075	Failed to read trigger width by USB
OPT_ERR_USBREADTRIGGERSRC_FAILED	3001076	Failed to read trigger source by USB
OPT_ERR_USBREADINTENSITY_FAILED	3001077	Failed to read intensity by USB
OPT_ERR_USBSETTRIGGERWIDTH_FAILED	3001078	Failed to set trigger width by USB
OPT_ERR_USBSETTRIGGERSRC_FAILED	3001079	Failed to set trigger source by USB
OPT_ERR_USBSETINTENSITY_FAILED	3001080	Failed to set intensity by USB

Note: for acronyms, please refer to Tab. 2

C Acronyms

Table 2: **Acronyms**

Acronym	Meaning
CH	channel
CON	connection
CONFIG	configuration
CUR	current
DHCP	Dynamic Host Configuration Protocol
ERR	error
ETHE	Ethernet
HB	high brightness
IP	Internet Protocol
MAC	media access control
PARAM	parameter
SN	serial number
UDP	User Datagram Protocol