

DCS_SDK

2.0.0

Generated by Doxygen 1.8.17

1 DCS-100 C++ SDK	1
1.1 Binary distribution	1
1.2 Example Code	1
2 Changelog	5
2.1 1.0.0	5
2.2 1.1.0	5
2.3 2.0.0	5
3 Namespace Index	7
3.1 Namespace List	7
4 Hierarchical Index	9
4.1 Class Hierarchy	9
5 Class Index	11
5.1 Class List	11
6 File Index	13
6.1 File List	13
7 Namespace Documentation	15
7.1 AdvancedIllumination Namespace Reference	15
7.1.1 Detailed Description	15
7.1.2 Enumeration Type Documentation	16
7.1.2.1 Channel	16
7.1.2.2 DCS_Type	16
7.1.2.3 Mode	16
7.1.2.4 Trigger	17
8 Class Documentation	19
8.1 AdvancedIllumination::DCS_100 Class Reference	19
8.1.1 Detailed Description	20
8.1.2 Constructor & Destructor Documentation	20
8.1.2.1 DCS_100() [1/2]	21
8.1.2.2 DCS_100() [2/2]	21
8.1.2.3 ~DCS_100()	21
8.1.3 Member Function Documentation	21
8.1.3.1 Channel()	21
8.1.3.2 ChannelCount()	22
8.1.3.3 connect()	22
8.1.3.4 disconnect()	22

8.1.3.5 firmwareVersion()	23
8.1.3.6 ipAddress() [1/2]	23
8.1.3.7 ipAddress() [2/2]	23
8.1.3.8 name() [1/2]	23
8.1.3.9 name() [2/2]	24
8.1.3.10 operator[]()	24
8.1.3.11 profileName() [1/2]	25
8.1.3.12 profileName() [2/2]	25
8.1.3.13 profileNames()	25
8.1.3.14 profileNumber() [1/2]	26
8.1.3.15 profileNumber() [2/2]	26
8.1.3.16 refreshConfig()	26
8.1.3.17 refreshProfiles()	27
8.1.3.18 saveProfile()	27
8.1.3.19 webConfigEnabled() [1/2]	27
8.1.3.20 webConfigEnabled() [2/2]	27
8.1.4 Friends And Related Function Documentation	28
8.1.4.1 DCS_Channel	28
8.2 AdvancedIllumination::DCS_100::DCS_Channel Class Reference	28
8.2.1 Detailed Description	29
8.2.2 Member Function Documentation	29
8.2.2.1 current() [1/2]	29
8.2.2.2 current() [2/2]	29
8.2.2.3 maxContinuous()	30
8.2.2.4 maxFrequency()	30
8.2.2.5 maxStrobe()	30
8.2.2.6 mode() [1/2]	31
8.2.2.7 mode() [2/2]	31
8.2.2.8 pulseDelay() [1/2]	31
8.2.2.9 pulseDelay() [2/2]	31
8.2.2.10 pulseWidth() [1/2]	32
8.2.2.11 pulseWidth() [2/2]	32
8.2.2.12 trigger()	33
8.2.2.13 triggerInput() [1/2]	33
8.2.2.14 triggerInput() [2/2]	33
8.2.2.15 triggerMode() [1/2]	33
8.2.2.16 triggerMode() [2/2]	34
8.2.3 Friends And Related Function Documentation	34
8.2.3.1 DCS_100	34

8.3 AdvancedIllumination::DCS_Info Class Reference	34
8.3.1 Detailed Description	35
8.3.2 Constructor & Destructor Documentation	35
8.3.2.1 DCS_Info()	35
8.3.2.2 ~DCS_Info()	36
8.3.3 Member Function Documentation	36
8.3.3.1 findAllInNetwork()	36
8.3.3.2 firmware()	36
8.3.3.3 host()	37
8.3.3.4 lighthouse()	37
8.3.3.5 name()	37
8.3.3.6 parse()	37
8.3.3.7 type()	38
8.4 AdvancedIllumination::DeviceError Class Reference	38
8.4.1 Detailed Description	39
8.4.2 Constructor & Destructor Documentation	39
8.4.2.1 DeviceError()	39
8.4.2.2 ~DeviceError()	39
8.4.3 Member Function Documentation	40
8.4.3.1 what()	40
8.5 AdvancedIllumination::DeviceWarning Class Reference	40
8.5.1 Detailed Description	41
8.5.2 Constructor & Destructor Documentation	41
8.5.2.1 DeviceWarning()	41
8.5.2.2 ~DeviceWarning()	41
8.5.3 Member Function Documentation	41
8.5.3.1 what()	42
9 File Documentation	43
9.1 sdk/build/readme.md File Reference	43
9.2 sdk/changelog.md File Reference	43
9.3 sdk/DCS100.cpp File Reference	43
9.3.1 Macro Definition Documentation	44
9.3.1.1 INVALID_SOCKET	44
9.3.1.2 MIN	44
9.3.1.3 SOCKET_ERROR	44
9.3.1.4 sprintf_s	44
9.3.2 Typedef Documentation	44
9.3.2.1 json	45

9.3.2.2 Socket_t	45
9.4 sdk/DCS100.h File Reference	45
9.4.1 Macro Definition Documentation	46
9.4.1.1 DCS_EXPORT	46
9.5 sdk/DCS_Info.cpp File Reference	46
9.5.1 Function Documentation	46
9.5.1.1 getIPString()	47
9.6 sdk/DCS_Info.h File Reference	47
9.6.1 Macro Definition Documentation	47
9.6.1.1 DCS_EXPORT	47
Index	49

Chapter 1

DCS-100 C++ SDK

This is a software development kit (SDK) for creating software that interacts with the Advanced illumination DCS-100E and DCS-103E devices. It is written using the 2011 ISO Standard for C++ (C++ 11).

1.1 Binary distribution

This release was compiled using the GNU toolchain version 9.3.0. Releases are available for Windows x86 and x64, and Linux x64.

See the [changelog](#) for more history.

1.2 Example Code

This example code queries each DCS device that is online and communicable, prints the ones it finds, and then connects to each and does the following:

For every channel:

- Sets channel to Continuous mode
- Pauses 200 milliseconds
- Sets the current on the channel to half of the light's continuous maximum
- Pauses 2 seconds
- Sets 0 current
- Sets the channel to Pulsed mode
- Sets the current on the channel to 1.2 A
- Sets pulse width to 10ms
- Sets pulse delay to 5us

- Sets trigger input to Input 3
- Pulses the light three times (500 ms between each)
- Sets the channel to 0 current

It then sets the device name to "Tested" and disconnects from the device.

```
#include "DCS100.h"
#include "DCS_Info.h"
#include <iostream>
#include <chrono>
#include <thread>
using namespace std;
namespace ai = AdvancedIllumination;
int main()
{
    auto list = ai::DCS_Info::findAllInNetwork(false);
    if(list.size() == 0) {
        cout << "No DCS units found.\n";
        return 0;
    }
    try
    {
        for (const ai::DCS_Info& dcs : list)
        {
            std::cout << "Found DCS: " << dcs.name() << " at " << dcs.host() << "\n";
            ai::DCS_100 device;
            device.connect(dcs.host());
            size_t N = device.ChannelCount();
            for (int i = 0; i < N; i++)
            {
                if (device[i].maxContinuous() == 0) continue;
                printf("Setting channel %d mode 1\n", i+1);
                device[i].mode(ai::Mode::Continuous);
                this_thread::sleep_for(chrono::milliseconds(200));
                double cc = 0.5*device[i].maxContinuous();
                printf("Setting channel %d current %.3f\n", i+1, cc);
                device[i].current(cc);
                this_thread::sleep_for(chrono::seconds(2));
                printf("Setting 0 current\n");
                device[i].current(0);
                printf("Setting strobe mode\n");
                device[i].mode(ai::Mode::Pulsed);
                printf("Setting 1.2A\n");
                device[i].current(1.2);
                printf("Setting PW 10ms\n");
                device[i].pulseWidth(10E-3);
                printf("Setting PD 5us\n");
                device[i].pulseDelay(5E-6);
                printf("Setting trigger input CH3\n");
                device[i].triggerInput(ai::Channel::Three);
                for (int j = 0; j < 3; j++)
                {
                    printf("(blink)\t");
                    device[i].trigger();
                    this_thread::sleep_for(chrono::milliseconds(500));
                }
                printf("\n");
                device[i].current(0);
                this_thread::sleep_for(chrono::milliseconds(500));
            }
            device.name("Tested");
            //device.ipAddress("10.1.103.50");
            device.webConfigEnabled(true);
            device.disconnect();
        }
    }
    catch (const ai::DeviceError& e)
    {
        cout << "Error: " << e.what() << "\n";
        return EXIT_FAILURE;
    }
    catch (const ai::DeviceWarning& e)
    {
        cout << "Warning: " << e.what() << "\n";
        return 1;
    }
    catch (const std::exception& e)
```



```
{  
    cout << "Exception: " << e.what() << "\n";  
    return 1;  
}  
return 0;  
}
```


Chapter 2

Changelog

2.1 1.0.0

Initial release. Depended on POCO external libraries.

2.2 1.1.0

Updated to support DCS-100E/103E RevA.

2.3 2.0.0

Major revision, removes all external dependencies, builds as static library. Includes TinyXML 2 for XML parsing (compiled in).

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

AdvancedIllumination	
Namespace containing AI-created functions and classes	15

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AdvancedIllumination::DCS_100	19
AdvancedIllumination::DCS_100::DCS_Channel	28
AdvancedIllumination::DCS_Info	34
invalid_argument	
AdvancedIllumination::DeviceWarning	40
runtime_error	
AdvancedIllumination::DeviceError	38

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AdvancedIllumination::DCS_100	
Contains information and methods for working with DCS devices	19
AdvancedIllumination::DCS_100::DCS_Channel	
Encapsulates the data related to a single DCS channel	28
AdvancedIllumination::DCS_Info	
Class containing information about a DCS-100 device	34
AdvancedIllumination::DeviceError	
An exception representing an error message from the device	38
AdvancedIllumination::DeviceWarning	
An exception representing a warning message from the device	40

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

sdk/ DCS100.cpp	43
sdk/ DCS100.h	45
sdk/ DCS_Info.cpp	46
sdk/ DCS_Info.h	47

Chapter 7

Namespace Documentation

7.1 AdvancedIllumination Namespace Reference

Namespace containing AI-created functions and classes.

Classes

- class [DCS_100](#)
Contains information and methods for working with DCS devices.
- class [DCS_Info](#)
Class containing information about a DCS-100 device.
- class [DeviceError](#)
An exception representing an error message from the device.
- class [DeviceWarning](#)
An exception representing a warning message from the device.

Enumerations

- enum [Mode](#) { [Mode::Off](#) = 0, [Mode::Continuous](#) = 1, [Mode::Pulsed](#) = 2, [Mode::Gated](#) = 3 }
Values that represent modes.
- enum [Trigger](#) { [Trigger::Falling](#) = 0, [Trigger::Rising](#) = 1 }
Values that represent trigger edges (e.g.
- enum [Channel](#) { [Channel::One](#) = 1, [Channel::Two](#) = 2, [Channel::Three](#) = 3 }
Values that represent channels.
- enum [DCS_Type](#) { [DCS_Type::DCS_100E](#), [DCS_Type::DCS_103E](#) }
Values that represents the type of DCS connected.

7.1.1 Detailed Description

Namespace containing AI-created functions and classes.

7.1.2 Enumeration Type Documentation

7.1.2.1 Channel

```
enum AdvancedIllumination::Channel [strong]
```

Values that represent channels.

Enumerator

One	The first channel.
Two	The second channel.
Three	The third channel.

7.1.2.2 DCS_Type

```
enum AdvancedIllumination::DCS_Type [strong]
```

Values that represents the type of DCS connected.

Enumerator

DCS_100E	An enum constant representing the DCS-100E (single output, three channel) device.
DCS_103E	An enum constant representing the DCS-103E (three output, single channel per output) device.

7.1.2.3 Mode

```
enum AdvancedIllumination::Mode [strong]
```

Values that represent modes.

Enumerator

Off	An enum constant representing the "Off" mode.
Continuous	An enum constant representing the Continuous current mode.
Pulsed	An enum constant representing the Pulsed mode.
Gated	An enum constant representing the Gated mode.

7.1.2.4 Trigger

```
enum AdvancedIllumination::Trigger [strong]
```

Values that represent trigger edges (e.g.

how each channel's triggering behaves).

Enumerator

Falling	A rising-edge trigger.
Rising	A falling-edge trigger.

Chapter 8

Class Documentation

8.1 AdvancedIllumination::DCS_100 Class Reference

Contains information and methods for working with DCS devices.

```
#include <DCS100.h>
```

Classes

- class [DCS_Channel](#)
Encapsulates the data related to a single DCS channel.

Public Member Functions

- [DCS_Channel](#) & [Channel](#) (size_t)
Gets a channel by its number.
- [DCS_Channel](#) & [operator\[\]](#) (size_t)
Array indexer operator.
- const std::string & [name](#) () const
Gets the name.
- void [name](#) (const std::string &name)
Sets the name of the device.
- const std::string & [firmwareVersion](#) () const
Gets the firmware version.
- const std::string & [ipAddress](#) () const
Get the IP address.
- void [ipAddress](#) (const std::string &ip)
Sets the device (static) IP address.
- size_t [ChannelCount](#) () const
Gets the channel count of the device.
- const std::string & [profileName](#) () const

- Gets the active profile name.*
 - int [profileNumber](#) () const noexcept
- Gets the active profile number.*
 - void [profileName](#) (const std::string &)
- Sets the name of the active profile.*
 - void [profileNumber](#) (int)
- Sets the active profile number.*
 - void [webConfigEnabled](#) (const bool)
- Enable or disable the Web configuration interface.*
 - bool [webConfigEnabled](#) () const
- Determine if the Web configuration interface is enabled.*
 - void [saveProfile](#) ()
- Saves the profile.*
 - [DCS_100](#) ()
- Default constructor.*
 - [DCS_100](#) (const char *ip)
- Creates an instance and connects to the given device.*
 - void [connect](#) (const std::string &ip)
- Connects to a DCS at the given IP address.*
 - void [disconnect](#) ()
- Disconnects this object from the physical DCS.*
 - void [refreshConfig](#) ()
- Refresh configuration from device.*
 - const std::vector< std::string > & [profileNames](#) () const
- Gets the names of the saved profiles.*
 - [~DCS_100](#) ()
- Destructor.*
 - void [refreshProfiles](#) ()
- Refresh profile names from device.*

Friends

- class [DCS_Channel](#)

8.1.1 Detailed Description

Contains information and methods for working with DCS devices.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 DCS_100() [1/2]

```
AdvancedIllumination::DCS_100::DCS_100 ( )
```

Default constructor.

8.1.2.2 DCS_100() [2/2]

```
AdvancedIllumination::DCS_100::DCS_100 (
    const char * device )
```

Creates an instance and connects to the given device.

Parameters

<i>device</i>	The device.
---------------	-------------

8.1.2.3 ~DCS_100()

```
AdvancedIllumination::DCS_100::~~DCS_100 ( )
```

Destructor.

8.1.3 Member Function Documentation

8.1.3.1 Channel()

```
DCS_100::DCS_Channel & AdvancedIllumination::DCS_100::Channel (
    size_t i )
```

Gets a channel by its number.

This is the one-based (1-3) complement of the array index operator, which is zero-based.

Exceptions

<i>invalid_argument</i>	Thrown when $i == 0$ or $i > 3$ (if built with bounds checking) (Third-party reference)
-------------------------	---

Parameters

<i>i</i>	One-based index of the channel.
----------	---------------------------------

Returns

A reference to a [DCS_100::DCS_Channel](#): the channel.

8.1.3.2 ChannelCount()

```
size_t AdvancedIllumination::DCS_100::ChannelCount ( ) const
```

Gets the channel count of the device.

Returns

The number of active channels in the device.

8.1.3.3 connect()

```
void AdvancedIllumination::DCS_100::connect (
    const std::string & ip )
```

Connects to a DCS at the given IP address.

Parameters

<i>ip</i>	The IP address of the device to connect to
-----------	--

8.1.3.4 disconnect()

```
void AdvancedIllumination::DCS_100::disconnect ( )
```

Disconnects this object from the physical DCS.

Allows the DCS to communicate with other SDK or software instances.

8.1.3.5 firmwareVersion()

```
const string & AdvancedIllumination::DCS_100::firmwareVersion ( ) const
```

Gets the firmware version.

Returns

A reference to a const string.

8.1.3.6 ipAddress() [1/2]

```
const string & AdvancedIllumination::DCS_100::ipAddress ( ) const
```

Get the IP address.

Returns

The device IP address

8.1.3.7 ipAddress() [2/2]

```
void AdvancedIllumination::DCS_100::ipAddress (
    const std::string & ip )
```

Sets the device (static) IP address.

Exceptions

<i>invalid_argument</i>	Thrown when the IP address string is invalid. (Third-party reference)
<i>DeviceWarning</i>	Thrown when the device response contains a warning.

Parameters

<i>ip</i>	The IP address to set (as a string).
-----------	--------------------------------------

8.1.3.8 name() [1/2]

```
const string & AdvancedIllumination::DCS_100::name ( ) const
```

Gets the name.

Returns

The device name.

8.1.3.9 name() [2/2]

```
void AdvancedIllumination::DCS_100::name (
    const std::string & name )
```

Sets the name of the device.

Exceptions

<i>invalid_argument</i>	Thrown when the name string is invalid. (Third-party reference)
<i>DeviceWarning</i>	Thrown when the device issues a warning.

Parameters

<i>name</i>	The name to set. Cannot contain commas.
-------------	---

8.1.3.10 operator[]()

```
DCS_100::DCS_Channel & AdvancedIllumination::DCS_100::operator[] (
    size_t i )
```

Array indexer operator.

Provides access to the channels of the DCS.

Exceptions

<i>invalid_argument</i>	Thrown when index > 2 (if built with bounds checking). (Third-party reference)
-------------------------	--

Parameters

<i>i</i>	Zero-based index of the channel to access.
----------	--

Returns

The indexed channel.

8.1.3.11 profileName() [1/2]

```
const string & AdvancedIllumination::DCS_100::profileName ( ) const
```

Gets the active profile name.

Returns

A reference to a const string.

8.1.3.12 profileName() [2/2]

```
void AdvancedIllumination::DCS_100::profileName (
    const std::string & name )
```

Sets the name of the active profile.

Exceptions

<i>DeviceWarning</i>	Thrown when a device warning error condition occurs.
<i>DeviceError</i>	Raised when a device error condition occurs.

Parameters

<i>name</i>	The name.
-------------	-----------

8.1.3.13 profileNames()

```
const std::vector< std::string > & AdvancedIllumination::DCS_100::profileNames ( ) const
```

Gets the names of the saved profiles.

Returns

A reference to a const std::vector of strings containing the profile names.

8.1.3.14 profileNumber() [1/2]

```
int AdvancedIllumination::DCS_100::profileNumber ( ) const [noexcept]
```

Gets the active profile number.

Returns

A const int.

8.1.3.15 profileNumber() [2/2]

```
void AdvancedIllumination::DCS_100::profileNumber (
    int num )
```

Sets the active profile number.

Parameters

<i>num</i>	The profile number (0-24) to set.
------------	-----------------------------------

8.1.3.16 refreshConfig()

```
void AdvancedIllumination::DCS_100::refreshConfig ( )
```

Refresh configuration from device.

8.1.3.17 refreshProfiles()

```
void AdvancedIllumination::DCS_100::refreshProfiles ( )
```

Refresh profile names from device.

8.1.3.18 saveProfile()

```
void AdvancedIllumination::DCS_100::saveProfile ( )
```

Saves the profile.

8.1.3.19 webConfigEnabled() [1/2]

```
bool AdvancedIllumination::DCS_100::webConfigEnabled ( ) const
```

Determine if the Web configuration interface is enabled.

Returns

A const bool, `true` for enabled and `false` for disabled.

Remarks

If the web configuration interface is enabled, a web browser may be directed to the [IP address](#) of the device.

8.1.3.20 webConfigEnabled() [2/2]

```
void AdvancedIllumination::DCS_100::webConfigEnabled (
    const bool b )
```

Enable or disable the Web configuration interface.

Parameters

<i>b</i>	<code>true</code> for enabled, <code>false</code> for disabled.
----------	---

8.1.4 Friends And Related Function Documentation

8.1.4.1 DCS_Channel

```
friend class DCS_Channel [friend]
```

The documentation for this class was generated from the following files:

- [sdk/DCS100.h](#)
- [sdk/DCS100.cpp](#)

8.2 AdvancedIllumination::DCS_100::DCS_Channel Class Reference

Encapsulates the data related to a single DCS channel.

```
#include <DCS100.h>
```

Public Member Functions

- double [current](#) () const
Gets the current.
- void [current](#) (const double)
Sets the current.
- double [pulseWidth](#) () const
Gets the pulse width.
- void [pulseWidth](#) (const double)
Sets the pulse width.
- double [pulseDelay](#) () const
Gets the pulse delay.
- void [pulseDelay](#) (const double)
Sets the pulse delay.
- double [maxContinuous](#) () const
Gets the maximum continuous current (per string of LEDs).
- const double [maxStrobe](#) () const
Gets the maximum strobe current (per string of LEDs).
- [Mode](#) [mode](#) () const
Gets the mode the channel is in.
- void [mode](#) (const [Mode](#))
Sets the channels' Mode.
- [Trigger](#) [triggerMode](#) () const
Gets the trigger mode (rising or falling edge).
- void [triggerMode](#) (const [Trigger](#))

- Sets the trigger mode.*
- [Channel triggerInput](#) () const
 - Gets the configured trigger input channel.*
- void [triggerInput](#) (const [Channel](#))
 - Sets the trigger input (e.g.*
- double [maxFrequency](#) () const
 - The maximum frequency (in Hertz) at which this channel can be triggered.*
- void [trigger](#) () const
 - Triggers this channel (acts as if a single hardware trigger occurred).*

Friends

- class [DCS_100](#)

8.2.1 Detailed Description

Encapsulates the data related to a single DCS channel.

Allows getting and setting the device current, pulse width, etc.

8.2.2 Member Function Documentation

8.2.2.1 [current\(\)](#) [1/2]

```
double AdvancedIllumination::DCS_100::DCS_Channel::current ( ) const
```

Gets the current.

Returns

The current (in Amps).

8.2.2.2 [current\(\)](#) [2/2]

```
void AdvancedIllumination::DCS_100::DCS_Channel::current (
    const double cur )
```

Sets the current.

Exceptions

<i>DeviceWarning</i>	Thrown when a device warning error condition occurs.
<i>DeviceError</i>	Raised when a Device error condition occurs.
<i>runtime_error</i>	Raised when a runtime error condition occurs. (Third-party reference)

Parameters

<i>cur</i>	The current (in Amps).
------------	------------------------

8.2.2.3 maxContinuous()

```
double AdvancedIllumination::DCS_100::DCS_Channel::maxContinuous ( ) const
```

Gets the maximum continuous current (per string of LEDs).

Returns

The maximum continuous current (per string), in Amps.

8.2.2.4 maxFrequency()

```
double AdvancedIllumination::DCS_100::DCS_Channel::maxFrequency ( ) const
```

The maximum frequency (in Hertz) at which this channel can be triggered.

Returns

A const double; the maximum trigger frequency for this channel.

8.2.2.5 maxStrobe()

```
const double AdvancedIllumination::DCS_100::DCS_Channel::maxStrobe ( ) const
```

Gets the maximum strobe current (per string of LEDs).

Returns

The maximum strobe current (per string), in Amps.

8.2.2.6 mode() [1/2]

```
Mode AdvancedIllumination::DCS_100::DCS_Channel::mode ( ) const
```

Gets the mode the channel is in.

Returns

The channel's Mode.

8.2.2.7 mode() [2/2]

```
void AdvancedIllumination::DCS_100::DCS_Channel::mode (
    const Mode )
```

Sets the channels' Mode.

Parameters

<i>m</i>	The Mode to set.
----------	------------------

8.2.2.8 pulseDelay() [1/2]

```
double AdvancedIllumination::DCS_100::DCS_Channel::pulseDelay ( ) const
```

Gets the pulse delay.

Returns

The pulse delay (in seconds).

8.2.2.9 pulseDelay() [2/2]

```
void AdvancedIllumination::DCS_100::DCS_Channel::pulseDelay (
    const double pd )
```

Sets the pulse delay.

Exceptions

<i>std::invalid_argument</i>	Thrown when pulse delay exceeds 10 milliseconds. (Third-party reference)
------------------------------	--

Parameters

<i>pd</i>	The pulse delay (in seconds).
-----------	-------------------------------

8.2.2.10 pulseWidth() [1/2]

```
double AdvancedIllumination::DCS_100::DCS_Channel::pulseWidth ( ) const
```

Gets the pulse width.

Returns

The pulse width (in seconds).

8.2.2.11 pulseWidth() [2/2]

```
void AdvancedIllumination::DCS_100::DCS_Channel::pulseWidth (
    const double pw )
```

Sets the pulse width.

Must be less than or equal to 65535 microseconds (0.065535 seconds)

Exceptions

<i>invalid_argument</i>	Thrown when pulse width exceeds 65535 microseconds. (Third-party reference)
<i>runtime_error</i>	Raised when a runtime error condition occurs. (Third-party reference)

Parameters

<i>pw</i>	The pulse width (in seconds).
-----------	-------------------------------

8.2.2.12 trigger()

```
void AdvancedIllumination::DCS_100::DCS_Channel::trigger ( ) const
```

Triggers this channel (acts as if a single hardware trigger occurred).

8.2.2.13 triggerInput() [1/2]

```
Channel AdvancedIllumination::DCS_100::DCS_Channel::triggerInput ( ) const
```

Gets the configured trigger input channel.

Returns

A const Channel.

8.2.2.14 triggerInput() [2/2]

```
void AdvancedIllumination::DCS_100::DCS_Channel::triggerInput (
    const Channel )
```

Sets the trigger input (e.g.

which hardware trigger input this channel is bound to).

Remarks

This function sets which hardware trigger input this channel responds to on trigger events. Any combination of channels can be set to any trigger input, however each channel can only be assigned to one hardware input. That is, all three channels on a DCS-100E can share trigger input 1, but channel one can't be triggered on inputs 1 and 2, only input 1 or 2 (not both).

Parameters

<i>chan</i>	The channel to which this channel's trigger will be assigned.
-------------	---

8.2.2.15 triggerMode() [1/2]

```
Trigger AdvancedIllumination::DCS_100::DCS_Channel::triggerMode ( ) const
```

Gets the trigger mode (rising or falling edge).

Returns

The channel's current trigger mode.

8.2.2.16 triggerMode() [2/2]

```
void AdvancedIllumination::DCS_100::DCS_Channel::triggerMode (
    const Trigger t )
```

Sets the trigger mode.

Exceptions

<i>DeviceWarning</i>	Thrown when a device warning error condition occurs.
<i>DeviceError</i>	Raised when a device error condition occurs.

Parameters

<i>t</i>	The trigger mode to set.
----------	--------------------------

8.2.3 Friends And Related Function Documentation

8.2.3.1 DCS_100

```
friend class DCS_100 [friend]
```

The documentation for this class was generated from the following files:

- sdk/[DCS100.h](#)
- sdk/[DCS100.cpp](#)

8.3 AdvancedIllumination::DCS_Info Class Reference

Class containing information about a DCS-100 device.

```
#include <DCS_Info.h>
```


Public Member Functions

- [DCS_Info](#) ()
Default constructor.
- [~DCS_Info](#) ()
Destructor.
- const std::string & [name](#) () const
Gets the name of the device.
- const std::string & [lighthouse](#) () const
Gets the part number of the connected lighthouse.
- const std::string & [firmware](#) () const
Gets the firmware version.
- const std::string & [host](#) () const
Gets the host name (IP address) of the device.
- [DCS_Type](#) [type](#) () const
Gets the type of DCS.

Static Public Member Functions

- static [DCS_Info](#) [parse](#) (const std::string &idn, std::string [host](#))
Parses the given IDN string.
- static std::vector< [DCS_Info](#) > [findAllInNetwork](#) (bool global=false)
Searches for all online DCS devices in the network.

8.3.1 Detailed Description

Class containing information about a DCS-100 device.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 DCS_Info()

```
AdvancedIllumination::DCS_Info::DCS_Info ( )
```

Default constructor.

8.3.2.2 ~DCS_Info()

```
AdvancedIllumination::DCS_Info::~~DCS_Info ( )
```

Destructor.

8.3.3 Member Function Documentation

8.3.3.1 findAllInNetwork()

```
vector< DCS_Info > AdvancedIllumination::DCS_Info::findAllInNetwork (
    bool global = false ) [static]
```

Searches for all online DCS devices in the network.

Parameters

<i>global</i>	Whether to use a global broadcast (255.255.255.255) or subnet (X.X.X.255)
---------------	---

Returns

The found all in network.

8.3.3.2 firmware()

```
const std::string & AdvancedIllumination::DCS_Info::firmware ( ) const
```

Gets the firmware version.

Returns

The device firmware version.

8.3.3.3 host()

```
const std::string & AdvancedIllumination::DCS_Info::host ( ) const
```

Gets the host name (IP address) of the device.

Returns

A reference to a const std::string, the device IP address.

8.3.3.4 lighthouse()

```
const std::string & AdvancedIllumination::DCS_Info::lighthouse ( ) const
```

Gets the part number of the connected lighthouse.

Returns

A reference to a const std::string. Empty if there is no light connected.

8.3.3.5 name()

```
const std::string & AdvancedIllumination::DCS_Info::name ( ) const
```

Gets the name of the device.

Returns

The device name.

8.3.3.6 parse()

```
DCS_Info AdvancedIllumination::DCS_Info::parse (
    const std::string & idn,
    std::string host ) [static]
```

Parses the given IDN string.

Exceptions

<i>invalid_argument</i>	Thrown when an invalid string is passed. (Third-party reference)
-------------------------	--

Parameters

<i>idn</i>	The IDN response string.
<i>host</i>	The host (IP address).

Returns

A [DCS_Info](#).

8.3.3.7 type()

[DCS_Type](#) `AdvancedIllumination::DCS_Info::type () const`

Gets the type of DCS.

Returns

A const [DCS_Type](#).

The documentation for this class was generated from the following files:

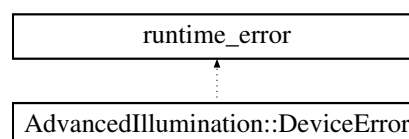
- [sdk/DCS_Info.h](#)
- [sdk/DCS_Info.cpp](#)

8.4 AdvancedIllumination::DeviceError Class Reference

An exception representing an error message from the device.

```
#include <DCS100.h>
```

Inheritance diagram for `AdvancedIllumination::DeviceError`:



Public Member Functions

- virtual const char * [what](#) () const noexcept
Gets the exception message.
- [DeviceError](#) (const std::string msg)
Constructor.
- virtual [~DeviceError](#) () noexcept
Destructor.

8.4.1 Detailed Description

An exception representing an error message from the device.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 DeviceError()

```
AdvancedIllumination::DeviceError::DeviceError (  
    const std::string msg )
```

Constructor.

Creates an instance of the exception with the given message.

Parameters

<i>msg</i>	The message.
------------	--------------

8.4.2.2 ~DeviceError()

```
AdvancedIllumination::DeviceError::~~DeviceError ( ) [virtual], [noexcept]
```

Destructor.

8.4.3 Member Function Documentation

8.4.3.1 what()

```
const char * AdvancedIllumination::DeviceError::what ( ) const [virtual], [noexcept]
```

Gets the exception message.

Returns

Null if it fails, else a pointer to a const char (C-string).

The documentation for this class was generated from the following files:

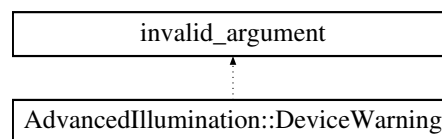
- [sdk/DCS100.h](#)
- [sdk/DCS100.cpp](#)

8.5 AdvancedIllumination::DeviceWarning Class Reference

An exception representing a warning message from the device.

```
#include <DCS100.h>
```

Inheritance diagram for AdvancedIllumination::DeviceWarning:



Public Member Functions

- virtual const char * [what](#) () const noexcept
Gets the exception message.
- [DeviceWarning](#) (const std::string msg)
Constructor.
- virtual [~DeviceWarning](#) () noexcept
Destructor.

8.5.1 Detailed Description

An exception representing a warning message from the device.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 DeviceWarning()

```
AdvancedIllumination::DeviceWarning::DeviceWarning (
    const std::string msg )
```

Constructor.

Creates an exception with the given message.

Parameters

<i>msg</i>	The message.
------------	--------------

8.5.2.2 ~DeviceWarning()

```
AdvancedIllumination::DeviceWarning::~~DeviceWarning ( ) [virtual], [noexcept]
```

Destructor.

8.5.3 Member Function Documentation

8.5.3.1 what()

```
const char * AdvancedIllumination::DeviceWarning::what ( ) const [virtual], [noexcept]
```

Gets the exception message.

Returns

Null if it fails, else a pointer to a const char (C-string).

The documentation for this class was generated from the following files:

- [sdk/DCS100.h](#)
- [sdk/DCS100.cpp](#)

Chapter 9

File Documentation

9.1 sdk/build/readme.md File Reference

9.2 sdk/changelog.md File Reference

9.3 sdk/DCS100.cpp File Reference

```
#include "DCS100.h"
#include "DCS_Info.h"
#include <cstdio>
#include "json.hpp"
#include "tinyxml2.h"
#include <regex>
#include <sstream>
#include <thread>
#include <chrono>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/select.h>
#include <netinet/in.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
```

Namespaces

- [AdvancedIllumination](#)

Namespace containing AI-created functions and classes.

Macros

- `#define sprintf_s sprintf`
- `#define SOCKET_ERROR (-1)`
- `#define INVALID_SOCKET (-1)`
- `#define MIN(a, b) ((a) < (b) ? (a) : (b))`

Typedefs

- `typedef int Socket_t`
- `using json = nlohmann::json`

9.3.1 Macro Definition Documentation

9.3.1.1 [INVALID_SOCKET](#)

```
#define INVALID_SOCKET (-1)
```

9.3.1.2 [MIN](#)

```
#define MIN(  
    a,  
    b ) ((a) < (b) ? (a) : (b))
```

9.3.1.3 [SOCKET_ERROR](#)

```
#define SOCKET_ERROR (-1)
```

9.3.1.4 [sprintf_s](#)

```
#define sprintf_s sprintf
```

9.3.2 Typedef Documentation

9.3.2.1 json

```
using json = nlohmann::json
```

9.3.2.2 Socket_t

```
typedef int Socket_t
```

9.4 sdk/DCS100.h File Reference

```
#include <string>
#include <cinttypes>
#include <vector>
#include <iostream>
#include <exception>
```

Classes

- class [AdvancedIllumination::DeviceWarning](#)
An exception representing a warning message from the device.
- class [AdvancedIllumination::DeviceError](#)
An exception representing an error message from the device.
- class [AdvancedIllumination::DCS_100](#)
Contains information and methods for working with DCS devices.
- class [AdvancedIllumination::DCS_100::DCS_Channel](#)
Encapsulates the data related to a single DCS channel.

Namespaces

- [AdvancedIllumination](#)
Namespace containing AI-created functions and classes.

Macros

- `#define` [DCS_EXPORT](#)

Enumerations

- enum [AdvancedIllumination::Mode](#) { [AdvancedIllumination::Mode::Off](#) = 0, [AdvancedIllumination::Mode::Continuous](#) = 1, [AdvancedIllumination::Mode::Pulsed](#) = 2, [AdvancedIllumination::Mode::Gated](#) = 3 }
- *Values that represent modes.*
- enum [AdvancedIllumination::Trigger](#) { [AdvancedIllumination::Trigger::Falling](#) = 0, [AdvancedIllumination::Trigger::Rising](#) = 1 }
- *Values that represent trigger edges (e.g.*
- enum [AdvancedIllumination::Channel](#) { [AdvancedIllumination::Channel::One](#) = 1, [AdvancedIllumination::Channel::Two](#) = 2, [AdvancedIllumination::Channel::Three](#) = 3 }
- *Values that represent channels.*

9.4.1 Macro Definition Documentation

9.4.1.1 DCS_EXPORT

```
#define DCS_EXPORT
```

9.5 sdk/DCS_Info.cpp File Reference

```
#include "DCS_Info.h"
#include <string>
#include <regex>
#include "findpulsars.h"
#include <sstream>
```

Namespaces

- [AdvancedIllumination](#)
- *Namespace containing AI-created functions and classes.*

Functions

- std::string [getIPString](#) (uint32_t ip)

9.5.1 Function Documentation

9.5.1.1 getIPString()

```
std::string getIPString (
    uint32_t ip )
```

9.6 sdk/DCS_Info.h File Reference

```
#include <vector>
#include <string>
#include "DCS100.h"
```

Classes

- class [AdvancedIllumination::DCS_Info](#)
Class containing information about a DCS-100 device.

Namespaces

- [AdvancedIllumination](#)
Namespace containing AI-created functions and classes.

Macros

- #define [DCS_EXPORT](#)

Enumerations

- enum [AdvancedIllumination::DCS_Type](#) { [AdvancedIllumination::DCS_Type::DCS_100E](#), [AdvancedIllumination::DCS_Type::DCS_1](#) }
- Values that represents the type of DCS connected.*

9.6.1 Macro Definition Documentation

9.6.1.1 DCS_EXPORT

```
#define DCS_EXPORT
```


Index

- ~DCS_100
 - AdvancedIllumination::DCS_100, [21](#)
- ~DCS_Info
 - AdvancedIllumination::DCS_Info, [35](#)
- ~DeviceError
 - AdvancedIllumination::DeviceError, [39](#)
- ~DeviceWarning
 - AdvancedIllumination::DeviceWarning, [41](#)
- AdvancedIllumination, [15](#)
 - Channel, [16](#)
 - Continuous, [16](#)
 - DCS_100E, [16](#)
 - DCS_103E, [16](#)
 - DCS_Type, [16](#)
 - Falling, [17](#)
 - Gated, [16](#)
 - Mode, [16](#)
 - Off, [16](#)
 - One, [16](#)
 - Pulsed, [16](#)
 - Rising, [17](#)
 - Three, [16](#)
 - Trigger, [17](#)
 - Two, [16](#)
- AdvancedIllumination::DCS_100, [19](#)
 - ~DCS_100, [21](#)
 - Channel, [21](#)
 - ChannelCount, [22](#)
 - connect, [22](#)
 - DCS_100, [20](#), [21](#)
 - DCS_Channel, [28](#)
 - disconnect, [22](#)
 - firmwareVersion, [22](#)
 - ipAddress, [23](#)
 - name, [23](#), [24](#)
 - operator[], [24](#)
 - profileName, [25](#)
 - profileNames, [25](#)
 - profileNumber, [26](#)
 - refreshConfig, [26](#)
 - refreshProfiles, [26](#)
 - saveProfile, [27](#)
 - webConfigEnabled, [27](#)
- AdvancedIllumination::DCS_100::DCS_Channel, [28](#)
 - current, [29](#)
 - DCS_100, [34](#)
 - maxContinuous, [30](#)
 - maxFrequency, [30](#)
 - maxStrobe, [30](#)
 - mode, [30](#), [31](#)
 - pulseDelay, [31](#)
 - pulseWidth, [32](#)
 - trigger, [32](#)
 - triggerInput, [33](#)
 - triggerMode, [33](#), [34](#)
- AdvancedIllumination::DCS_Info, [34](#)
 - ~DCS_Info, [35](#)
 - DCS_Info, [35](#)
 - findAllInNetwork, [36](#)
 - firmware, [36](#)
 - host, [36](#)
 - lighthouse, [37](#)
 - name, [37](#)
 - parse, [37](#)
 - type, [38](#)
- AdvancedIllumination::DeviceError, [38](#)
 - ~DeviceError, [39](#)
 - DeviceError, [39](#)
 - what, [40](#)
- AdvancedIllumination::DeviceWarning, [40](#)
 - ~DeviceWarning, [41](#)
 - DeviceWarning, [41](#)
 - what, [41](#)
- Channel
 - AdvancedIllumination, [16](#)
 - AdvancedIllumination::DCS_100, [21](#)
- ChannelCount
 - AdvancedIllumination::DCS_100, [22](#)
- connect
 - AdvancedIllumination::DCS_100, [22](#)
- Continuous
 - AdvancedIllumination, [16](#)
- current
 - AdvancedIllumination::DCS_100::DCS_Channel, [29](#)
- DCS100.cpp
 - INVALID_SOCKET, [44](#)
 - json, [44](#)
 - MIN, [44](#)
 - SOCKET_ERROR, [44](#)

- Socket_t, 45
- sprintf_s, 44
- DCS100.h
 - DCS_EXPORT, 46
- DCS_100
 - AdvancedIllumination::DCS_100, 20, 21
 - AdvancedIllumination::DCS_100::DCS_Channel, 34
- DCS_100E
 - AdvancedIllumination, 16
- DCS_103E
 - AdvancedIllumination, 16
- DCS_Channel
 - AdvancedIllumination::DCS_100, 28
- DCS_EXPORT
 - DCS100.h, 46
 - DCS_Info.h, 47
- DCS_Info
 - AdvancedIllumination::DCS_Info, 35
- DCS_Info.cpp
 - getIPString, 46
- DCS_Info.h
 - DCS_EXPORT, 47
- DCS_Type
 - AdvancedIllumination, 16
- DeviceError
 - AdvancedIllumination::DeviceError, 39
- DeviceWarning
 - AdvancedIllumination::DeviceWarning, 41
- disconnect
 - AdvancedIllumination::DCS_100, 22
- Falling
 - AdvancedIllumination, 17
- findAllInNetwork
 - AdvancedIllumination::DCS_Info, 36
- firmware
 - AdvancedIllumination::DCS_Info, 36
- firmwareVersion
 - AdvancedIllumination::DCS_100, 22
- Gated
 - AdvancedIllumination, 16
- getIPString
 - DCS_Info.cpp, 46
- host
 - AdvancedIllumination::DCS_Info, 36
- INVALID_SOCKET
 - DCS100.cpp, 44
- ipAddress
 - AdvancedIllumination::DCS_100, 23
- json
 - DCS100.cpp, 44
- lighthouse
 - AdvancedIllumination::DCS_Info, 37
- maxContinuous
 - AdvancedIllumination::DCS_100::DCS_Channel, 30
- maxFrequency
 - AdvancedIllumination::DCS_100::DCS_Channel, 30
- maxStrobe
 - AdvancedIllumination::DCS_100::DCS_Channel, 30
- MIN
 - DCS100.cpp, 44
- Mode
 - AdvancedIllumination, 16
- mode
 - AdvancedIllumination::DCS_100::DCS_Channel, 30, 31
- name
 - AdvancedIllumination::DCS_100, 23, 24
 - AdvancedIllumination::DCS_Info, 37
- Off
 - AdvancedIllumination, 16
- One
 - AdvancedIllumination, 16
- operator[]
 - AdvancedIllumination::DCS_100, 24
- parse
 - AdvancedIllumination::DCS_Info, 37
- profileName
 - AdvancedIllumination::DCS_100, 25
- profileNames
 - AdvancedIllumination::DCS_100, 25
- profileNumber
 - AdvancedIllumination::DCS_100, 26
- Pulsed
 - AdvancedIllumination, 16
- pulseDelay
 - AdvancedIllumination::DCS_100::DCS_Channel, 31
- pulseWidth
 - AdvancedIllumination::DCS_100::DCS_Channel, 32
- refreshConfig
 - AdvancedIllumination::DCS_100, 26
- refreshProfiles
 - AdvancedIllumination::DCS_100, 26
- Rising
 - AdvancedIllumination, 17
- saveProfile
 - AdvancedIllumination::DCS_100, 27
- sdk/build/readme.md, 43
- sdk/changelog.md, 43
- sdk/DCS100.cpp, 43

- sdk/DCS100.h, [45](#)
- sdk/DCS_Info.cpp, [46](#)
- sdk/DCS_Info.h, [47](#)
- SOCKET_ERROR
 - DCS100.cpp, [44](#)
- Socket_t
 - DCS100.cpp, [45](#)
- sprintf_s
 - DCS100.cpp, [44](#)

- Three
 - AdvancedIllumination, [16](#)
- Trigger
 - AdvancedIllumination, [17](#)
- trigger
 - AdvancedIllumination::DCS_100::DCS_Channel, [32](#)
- triggerInput
 - AdvancedIllumination::DCS_100::DCS_Channel, [33](#)
- triggerMode
 - AdvancedIllumination::DCS_100::DCS_Channel, [33](#), [34](#)

- Two
 - AdvancedIllumination, [16](#)
- type
 - AdvancedIllumination::DCS_Info, [38](#)

- webConfigEnabled
 - AdvancedIllumination::DCS_100, [27](#)
- what
 - AdvancedIllumination::DeviceError, [40](#)
 - AdvancedIllumination::DeviceWarning, [41](#)