
# Programmierhandbuch

# S-SDK-MSC15



Dokument erstellt: 9.9.2021

# Kapitel 1

# Informationen

Gigahertz-Optik

Bitte lesen Sie die diese Dokumentation und den Haftungsausschluss vor Benutzung der Software sorgfältig durch.
**Mit der Installation und der Nutzung der Software erkennen Sie diese ausdrücklich in allen Teilen an.**
Die Firma Gigahertz-Optik GmbH behält sich das Recht vor, Änderungen an dieser Anleitung jederzeit und ohne Vorankündigung durchführen zu können.

## 1.1 Haftungsausschluss

Diese Software wurde mit größter Sorgfalt entwickelt und auf verschiedenen Rechnersystemen sorgfältig getestet. Dabei waren für die freigegebenen Produktversionen keine Fehler festzustellen. Es kann aber nicht garantiert werden, dass die Software auf jedem Zielsystem hundertprozentig fehlerfrei läuft. Eine vollständig fehlerfreie Software ist nach dem heutigen Stand der Technik nicht möglich.
Gigahertz-Optik GmbH übernimmt keine Gewähr dafür, dass die Software für die von Ihnen bestimmten Zwecke, für die Sie die Software einsetzen wollen, tauglich ist oder mit anderer, von Ihnen gewählter Software kompatibel ist. Sie tragen die alleinige Verantwortung für Auswahl, Installation und Nutzung sowie für die damit beabsichtigten Ergebnisse.
Mit Ausnahme von vorsätzlich verursachten Schäden haftet Gigahertz-Optik GmbH nicht für irgendeinen Schaden, der durch die Verwendung oder die Unmöglichkeit der Verwendung der Software verursacht worden ist. Dies gilt ohne Ausnahme auch für entgangenen Geschäftsgewinn, Betriebsunterbrechungen, entgangene Geschäftsinformation oder anderen wirtschaftlichen Verlust, auch wenn Gigahertz-Optik GmbH vorher auf die Möglichkeit eines solchen Schadens hingewiesen wurde. Die beiliegende Dokumentation/Hilfe der Software erhebt keinen Anspruch auf Richtigkeit und Vollständigkeit.

## 1.2 Garantie

Die Firma Gigahertz-Optik GmbH garantiert, dass sämtliche in dieser Produktbeschreibung aufgeführten Funktionen ausgeliefert werden. Auslieferungsmedien, falls vorhanden, sind frei von Materialfehlern.
Wir haben alle möglichen Schritte unternommen, um diese Software frei von Viren, Spyware, sogenannten "Back Door Entrances" oder anderen schädlichen Code zu halten. Wir sammeln keine Informationen über Sie oder Ihre Daten. Wir werden Ihnen nicht vorsätzlich die Möglichkeit entziehen, die Funktionen dieser Software zu nutzen oder auf Ihre Daten zuzugreifen. Diese Vereinbarung ersetzt nicht vertragliche Zusicherungen, die wir Ihnen gegenüber erklärt haben. Jede Veränderung an dieser Vereinbarung muss von beiden Parteien schriftlich bestätigt werden.

## 1.3 Lizenz

Eine Lizenz der Vollversion gestattet die Benutzung des Produkts auf genau einem Rechner. Jede gleichzeitige Benutzung auf einem weiteren Rechner erfordert eine zusätzliche Produktlizenz. Der Verbreitung unserer Prokukte oder Dokumentation ist nicht gestattet. Sie sind berechtigt, eine Kopie des Produktes zu Sicherungszwecken (Backup) anzufertigen. Sie sind berechtigt, eigene Software, die mit Hilfe dieses Entwicklungspakets angefertigt wurde, zusammen mit den benötigten DLLs dieses Entwicklungspakets weiterzugeben.

## 1.4 Überblick

Dieses Entwicklungspaket liefert Ihnen alle benötigten Hilfsmittel (keine Compiler oder integrierte Softwareentwicklungsumgebungen), die Sie benötigen, um ein Messgerät der Serie MSC15 & CSS Gerätevariante der Firma Gigahertz-Optik GmbH mit Hilfe von C/C++ direkt ansteuern zu können. Dies betrifft in erster Linie Kommunikationsbibliotheken und Steuerungsbibliotheken für Ihr MSC15 & CSS Gerätevariante.
Um diese Bibliotheken nutzen zu können, benötigen Sie eine Programmierumgebung, wie z.B. Mircosoft Visual Studio oder Embarcadero C++ Builder oder ähnliches.

## 1.5 Kontaktdaten der Firma Gigahertz-Optik

| Gigahertz Optik GmbH | Gigahertz-Optik Inc |
|---|---|
| An der Kälberweide 12<br>D-82299 Türkenfeld Germany<br>Tel.: +49 8193 93700-0<br>Fax: +49 8193 93700-50<br>Email: info@gigahertz-optik.de<br>Homepage:<br>http://www.gigahertz-optik.de | 110 Haverhill Road<br>Amesbury MA 01913 USA<br>Tel: + 978 462 1818<br>Fax: + 978 462 3677<br>Email: b.angelo@gigahertz-optik.com<br>Homepage:<br>https://www.gigahertz-optik.com |

## 1.6 Anforderungen

Für die Benutzung des MSC15 & CSS Gerätevariante-SDK müssen Sie folgende Systemanforderungen beachten:

- Minimaler Festplattenplatz ca. 10MB
- Betriebssystem: MS Windows 7 (32bit/64bit), MS Windows 10 (32bit/64bit)
- C/C++ Entwicklungsumgebungen wie beispielsweise MS Visual Studio, Embarcadero C++ Builder, ... when programming with C/C++
- freier USB-Anschluss

## 1.7 Installation

Zur Installation des MSC15 & CSS Gerätevariante-SDK von der Produkt-CD gehen Sie folgendermaßen vor:

- Lesen Sie diese Dokumentation, bevor Sie mit der Installation beginnen.

- Schließen Sie alle anderen Anwendungen vor der Installation.
- Legen Sie die CD in Ihr CD-Laufwerk oder entpacken Sie die ausgelieferte ZIP-Datei.
- Kopieren Sie den Gigahertz-Optik Ordner von der CD oder der ZIP-Datei an einen Ort Ihrer Wahl. Wenn Sie bereits andere Gigahertz-Optik Entwicklungspakete für andere Geräte der Firma Gigahertz-Optik GmbH installiert haben, benutzen Sie bitte denselben Installationspfad, um mögliche Konflikte zu vermeiden.
- Fügen Sie den Ordner "install dir/Gigahertz-Optik/runtime"ť Ihrem Systempfad hinzu. "install dir" entspricht hierbei dem Basispfad, in welchen Sie im Schritt 4 die Installation durchgeführt haben. Wenn Sie bereits andere Entwicklungspakete der Firma Gigahertz-Optik GmbH installiert in Gebrauch haben, kann Schritt 5 entfallen.

## 1.8 Systemvorbereitungen

Verbinden Sie das MSC15 & CSS Gerätevariante mit Ihrem Computers. Die benötigten Treiber sind Standardtreiber von Windows und werden automatisch installiert.
Dieses SDK ist passwortgeschützt! Jedes Programm, das mit diesem SDK erstellt wird, muss die Funktion set↩ Passwort() aufrufen, bevor irgendeine andere Funktion verwendet werden kann.

# Kapitel 2

# Programmierbeispiele

Beispiele zum Einbinden des SDKs in die Applikation

**Zu beachten**

Unter den einzelenen Kapitel der Methodenbeschreibungen (Moduls) befinden sich seperate Beispiele wie die Methode zu verwenden sind.

## 2.1 C++

Da wir keine Bibliotheken zum Import in unterschiedliche Entwicklungsumgebungen anbieten, muss zur Laufzeit dynamisch gelinkt werden, um alle zur Verfügung stehenden Methoden der DLL zu verwenden.
Dieses Beispiel zeigt den Import ausgewählter DLL - Methoden und Verwendung in einer C++ Klasse. Das Beispiel enthält nicht sämtliche zur Verfügung stehenden Methoden. Die Verwendung des handles wird in der Klasse gekapselt. Das Beispiel sucht und initialisiert ein MSC15 & CSS Gerätevariante, führt eine Messung durch und schreibt die Ergebnisse auf die Konsole.
Am Ende werden alle MSC15 & CSS Gerätevariante - Resourcen wieder freigegeben.

### 2.1.1 MSC15Example.cpp

```
#include "MSC15Import.h
#include <iostream>
int main(int argc, char* argv[])
{
    MSC15Import msc15;
    //search for a MSC15 device
    //first you have to replace the right password in the msc15Import.cpp
    int error = msc15.init("MSC15_0
);
    if (error == 0)
    {
        //set measurement mode and start a new measurement
        error = msc15.measure();
        //if no error occured read the integral values
        if (error == 0)
        {
            double value;
            msc15.getPhotopic(&value);
            std::cout « Ë(V):
 « value « std::endl;
            msc15.getCCT(&value);
            std::cout « "CCT:
 « value « std::endl;
        }
        else
        {
            std::cout « ërror occured:
```

```
 « error « std::endl;
        }
        msc15.close();
    }
    else
    {
        std::cout « ërror occured:
 « error « std::endl;
    }
    system("PAUSE
);
}
```

### 2.1.2 MSC15Import.cpp

```
#include "MSC15Import.h
MSC15Import::MSC15Import()
{
    hDLLGOMSC15 = NULL;
    handle = -1;
}
MSC15Import:: MSC15Import()
{
}
int __stdcall MSC15Import::init(char* deviceName)
{
    int l_rc = 0;
    if (handle > 0)
        close();
    if (getProcAddresses(&hDLLGOMSC15, "GOMDMSC15.dll
, 6,
        &GOMDMSC15_setPassword, "GOMDMSC15_setPassword
,
        &GOMDMSC15_getHandle, "GOMDMSC15_getHandle
,
        &GOMDMSC15_releaseHandle, "GOMDMSC15_releaseHandle
,
        &GOMDMSC15_measure, "GOMDMSC15_measure
,
        &GOMDMSC15_getCCT, "GOMDMSC15_getCCT
,
        &GOMDMSC15_getPhotopic, "GOMDMSC15_getPhotopic
        ))
    {
        try {
            l_rc = GOMDMSC15_setPassword("passw
); //replace passw with the right password
            if (l_rc == 0)
                l_rc = GOMDMSC15_getHandle(deviceName, &handle);
            if (handle > 0)
                std::cout « Ïnitialisation sucessfull
 « std::endl;
        }
        catch (...) {
            l_rc = -1;
        }
    }
    else {
        l_rc = -1;
    }
    return l_rc;
}
int __stdcall MSC15Import::measure()
{
    int l_rc = GOMDMSC15_measure(handle);
    return l_rc;
}
int __stdcall MSC15Import::getPhotopic(double* value)
{
    int l_rc = GOMDMSC15_getPhotopic(handle, value);
    return l_rc;
}
int __stdcall MSC15Import::getCCT(double* value)
{
    int l_rc = GOMDMSC15_getCCT(handle, value);
    return l_rc;
}
int __stdcall MSC15Import::close()
{
    int l_rc = GOMDMSC15_releaseHandle(handle);
    handle = -1;
    return l_rc;
}
bool __stdcall MSC15Import::getProcAddresses(HINSTANCE *p_hLibrary,
```

```
    const char* p_dllName, INT p_count, ...)
{
    va_list l_va;
    va_start(l_va, p_count);
    if ((*p_hLibrary = LoadLibrary(p_dllName)) != NULL)
    {
        FARPROC* l_procFunction = NULL;
        char* l_funcName = NULL;
        int l_idxCount = 0;
        while (l_idxCount < p_count)
        {
            l_procFunction = va_arg(l_va, FARPROC*);
            l_funcName = va_arg(l_va, LPSTR);
            if ((*l_procFunction =
                GetProcAddress(*p_hLibrary, l_funcName)) == NULL)
            {
                l_procFunction = NULL;
                return FALSE;
            }
            l_idxCount++;
        }
    }
    else
    {
        va_end(l_va);
        return false;
    }
    va_end(l_va);
    return true;
}
```

### 2.1.3 MSC15Import.h

```
#ifndef MSC15ImportH
#define MSC15ImportH
#include <Windows.h>
#include ßtdio.h
#include <iostream>
class MSC15Import
{
public:
    MSC15Import();
    virtual  MSC15Import();
    int __stdcall init(char* deviceName);
    int __stdcall close();
    int __stdcall getPhotopic(double* value);
    int __stdcall getCCT(double* value);
    int __stdcall measure();
private:
    int handle;
    HINSTANCE hDLLGOMSC15;
    bool __stdcall getProcAddresses(HINSTANCE *p_hLibrary, const char* p_dllName, int p_count, ...);
    int(__stdcall *GOMDMSC15_setPassword)(char* value);
    int(__stdcall *GOMDMSC15_getHandle)(char* device, int* handle);
    int(__stdcall *GOMDMSC15_releaseHandle)(int handle);
    int(__stdcall *GOMDMSC15_measure)(int handle);
    int(__stdcall *GOMDMSC15_getCCT)(int handle, double* value);
    int(__stdcall *GOMDMSC15_getPhotopic)(int handle, double* value);
};
#endif
```

## 2.2 Weitere Beispiel

Weitere Beispiele zum Einbinden der DLL's, befinden sich im Installationsverzeichnis der SDK.

# Kapitel 3

# Versionsübersicht

Eine Versionsübersicht des S-SDK-MSC15 folgt:

- **V2016.1**: Neu: Release des SDK
- **V2017.1**: Neu: Unterstützung für MSC15-W Geräte/ Neu: TM-30-15/ Update: Bilirubin und Melanopic auf die neuen Standards
- **V2017.2**: Neu: Spektrum in pixelbasierten Schritten/ Update: maximale Messzeit der Pulsmessung auf die Abtastrate angepasst
- **V2018.1**: Neu: Unterstützung des CSS-45
- **V2019.1**: Neu: Laden von Messwerten aus dem Geräte Logger/ Neu: Wellenlängenkorrektur
- **V2019.2**: Bugfix: Farbberechnung
- **V2019.3**: Neu: Unterstützung des CSS-45-D, Bugfix: RS485 Kommunikation
- **V2019.4**: Update: TM-30-15 auf TM-30-18, Bugfix: Funktionsnamen in der Schnittstelle
- **V2019.5**: New: GOMDMSC15_measureWithManualIntegrationTime(), Bugfix: Fehler bei Firmware $>$ 1.45
- **V2019.6**: Bugfix: GOMDMSC15_getLastIntegrationTime(), Neu: GOMDMSC15_getTemperature()
- **V2019.7**: Update: Die Berechnung der Messwerte findet jetzt in der DLL statt (schneller)
- **V2020.1**: Update: Melanopsin nach CIE S026
- **V2020.2**: Neu: Unterstützung des MSC15-Bili V01
- **V2020.3**: Update: Benutzerspezifische WL-Bereiche des MSC15-W
- **V2020.4**: Update: Ausgabe Messfehler wenig/kein Signal
- **V2021.1**: Update: RS485 Baudrate des CSS-45/ Bugfix: Interner Logger und RTC
- **V2021.2**: Neu: komplemetäre Wellenlänge/ Bugfix: Dominante Wellenlänge und CRI Berechnung

# Kapitel 4

# Errors and Warnings

Eine Auflistung der Fehlercodes und Warnungen folgt:

## 4.1 Fehler

- -500: Communication Error
- -25000: Kommunikationsproblem
- -25001: Setup Datei ungültig für device
- -25002: Setup Datei konnte nicht geöffnet werden
- -25004: Kein gültiger Handle
- -25005: Kommunikationskanal kann nicht initialisiert werden
- -25006: zu niedrige Firmwareversion
- -25007: Problem beim Daten Senden
- -25008: Problem beim Daten Empfangen
- -25009: Gerät sendet einen nicht näher definierten Fehler
- -25014: error main data eeprom
- -25015: error color data eeprom
- -25016: error correction factor data eeprom
- -25017: error dark value eeprom
- -25018: error calibration factor eeprom
- -25020: zu viel Umgebungslicht
- -25021: Messung wurde abgebrochen
- -25024: error user data eeprom
- -25025: die spektrale einheit meldet zu wenig Signal
- -25031: Wrong device type, method not valid
- -25032: es wurde ein falsches Passwort übergeben
- -25041: Device Type CSS-D without detector not valid for SDK
- -25100: Wert außerhalb des Gültigkeitbereichs

## 4.2 Warnungen

- 25003: file not found: es wurden bislang keine Defaultdaten gespeichert. Daher existiert auch keine Defaultdatei
- 25010: Batterie niedrig
- 25022: unstabiles Signal
- 25023: die spektrale Einheit meldet einen “overload”

# Kapitel 5

# Modul-Dokumentation

## 5.1 Information zu den S-SDK-MSC15 Methoden

Allgemeine Information, welche die GOMDMSC15 DLL betreffen.

Alle hier beschrieben Methoden können bei jedem MSC15 & CSS Gerätevariante angewendet werden. Die unterstützten Gerätevarianten sehen Sie in der Funktion GOMDMSC15_getMSC15DeviceType(). Abhängig von der Konfiguration, Kalibrierung und Ausstattung Ihres Messgerätes können sich gewisse Unterschiede in der Verwendung ergeben, so dass manche Methoden bei spezifischen Gerätekonfigurationen eventuell kein Resultat liefern.
Jede Methode liefert einen Rückgabewert. Rückgabewert „0“ bedeutet eine fehlerfreie Abarbeitung der Methode. Werte kleiner „0“ kennzeichnen das Auftreten eines Fehlers. Werte größer „0“ sind hingegen nur als Warnungen zu werten.
Eine Liste aller Rückgabewerte befindet sich am Ende der Dokumentation. In jedem Kapitel befindet sich meist ein Aufrufbeispiel wie wie Sie die Methodengruppe verwenden und einsetzen können. Zudem enthält die Dokumentation ein Beispiel wie Sie das SDK in Ihre Applikation einbinden können.

## 5.2 Standard SDK Methoden

### Funktionen

- int __stdcall GOMDMSC15_setPassword (char *value)
- int __stdcall GOMDMSC15_getDLLVersion (char *value)
- int __stdcall GOMDMSC15_getHandle (char *deviceName, int *handle)
- int __stdcall GOMDMSC15_releaseHandle (int handle)
- int __stdcall GOMDMSC15_getSerialNumber (int handle, char *value)
- int __stdcall GOMDMSC15_getFirmwareVersion (int handle, double *value)
- int __stdcall GOMDMSC15_isConnected (int handle, bool *value)
- int __stdcall GOMDMSC15_readStatus (int handle, int *status)
- int __stdcall GOMDMSC15_getMSC15DeviceType (int handle, int *type)
- int __stdcall GOMDMSC15_getDetectorType (int handle, int *type)
- int __stdcall GOMDMSC15_getDetectorSerialNumber (int handle, char *value)
- int __stdcall GOMDMSC15_getTemperature (int handle, double *value)

### 5.2.1 Ausführliche Beschreibung

Methoden für die Initalisierung des MSC15 & CSS Gerätevariante und allgemeinen Verwendung des SDKs.

### 5.2.2 C++ Aufrufbeispiel

MSC15 & CSS Gerätevariante-Gerät mit dem SDK initialisieren.

```
GOMDMSC15_setPassword("Your password
");
int handle;
int l_rc = GOMDMSC15_getHandle("MSC15_0
", &handle);     // initialization of first found device
if (handle > 0 )
{
    // do something
}
GOMDMSC15_releaseHandle(handle);                        // re-initialization
```

### 5.2.3 Dokumentation der Funktionen

#### 5.2.3.1 GOMDMSC15_setPassword()

```
int __stdcall GOMDMSC15_setPassword (
              char * value )
```

Diese Methode muss vor allen anderen aufgerufen werden, um die Benutzung des SDK freizuschalten. Die Freischaltung erfolgt auf mehreren Ebenen.
Ebene1: Benutzung des SDK im Allgemeinen
Ebene2: Alle Elemente der 1. Ebene plus die Speicherung von Kalibrierungen im benutzerspezifischen Speicherbereich
Die Passworte werden Ihnen je von der Firma Gigahertz-Optik GmbH gesondert mitgeteilt.

**Parameter**

| | | |
|---|---|---|
| in | *value* | Nullterminierter String, der das Passwort beinhaltet. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.2 GOMDMSC15_getDLLVersion()

```
int __stdcall GOMDMSC15_getDLLVersion (
             char * value )
```

Liefert die Versionsnummer dieser DLL.

**Parameter**

| | | |
|---|---|---|
| out | *value* | Nullterminierter String; enthält nach Rücksprung die Versionsnummer, Mindestgröße: 10 Bytes |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.3 GOMDMSC15_getHandle()

```
int __stdcall GOMDMSC15_getHandle (
             char * deviceName,
             int * handle )
```

Nach der Freischaltung des SDK muss grundsätzlich diese Methode als nächstes aufgerufen werden, um das MSC15 zu initialisieren. Der Parameter „handle“ beinhaltet eine eindeutige Zuordnungsnummer zum instanziierten Messgerät, die beim Aufruf aller weiteren Methoden als erster Parameter übergeben werden muss.

**Parameter**

| | | |
|---|---|---|
| in | *deviceName* | Nullterminierter String, der das gewünschte zu initialisierende Gerät kennzeichnet. Der String hat immer folgenden Aufbau: “MSC15_<serial>” oder or “CSS45_<serial>”. <serial> steht als Platzhalter für die Seriennummer des Messgeräts. Der String „MSC15_5678“ initialisiert z.B. das MSC15 mit der Seriennummer 5678. Eine weitere Möglichkeit ist die Übergabe von "MSC15_0". Damit wird das erste unter Windows registrierte MSC15 initialisiert. Der nächste Aufruf von getHandle("MSC15_0", &handle), liefert das nächste angeschlossene MSC15/CSS-45, sofern der Handle des ersten Geräts nicht wieder freigegeben wurde. |
| out | *handle* | Pointer auf einen Integer Wert; dieser Wert beinhaltet nach Rücksprung einen Handle > 0, wenn die Initialisierung erfolgreich war, ansonsten 0. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.4 GOMDMSC15_releaseHandle()

```
int __stdcall GOMDMSC15_releaseHandle (
            int handle )
```

Diese Methode muss zum Abschluss aufgerufen werden, um die vom MSC15/CSS-45 belegten Resourcen / Speicher wieder freizugeben.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.5 GOMDMSC15_getSerialNumber()

```
int __stdcall GOMDMSC15_getSerialNumber (
            int handle,
            char * value )
```

Liefert die Seriennummer des verbundenen Geräts. Wenn Sie ein CSS-D + CSS-45 benutzen sehen Sie auch GOMDMSC15_getDetectorSerialNumber().

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Nullterminierter String; enthält nach Rücksprung die Seriennummer des Geräts, Mindestgröße: 10 Bytes |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.6 GOMDMSC15_getFirmwareVersion()

```
int __stdcall GOMDMSC15_getFirmwareVersion (
            int handle,
            double * value )
```

Liefert die Firmware-Version des direkt mit dem PC verbundenen Geräts. Beispiel: CSS-D + CSS-45, liefert die FW Version vom CSS-D.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Nullterminierter String; enthält nach Rücksprung die Firmware-Version, Mindestgröße: 10 Bytes |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.7 GOMDMSC15_isConnected()

```
int __stdcall GOMDMSC15_isConnected (
            int handle,
            bool * value )
```

Ruft ab ob das Gerät mit dem aktuellen Handle noch verbunden ist.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Boolean; gibt an ob das MSC15 & CSS Gerätevariante verbunden ist:<br>• true: MSC15 & CSS Gerätevariante ist verbunden<br>• false: MSC15 & CSS Gerätevariante ist nicht verbunden |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.8 GOMDMSC15_readStatus()

```
int __stdcall GOMDMSC15_readStatus (
            int handle,
            int * status )
```

Ruft den aktuellen Geräte-Status des MSC15/CSS-45 ab. Bei Werten ungleich "0" siehe die Rückgabewerttabelle (Fehler & Warnungen).

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *status* | Integer Wert; bei Werten ungleich „0“ siehe Rückgabewerttabelle (Fehler & Warnungen). |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.9 GOMDMSC15_getMSC15DeviceType()

```
int __stdcall GOMDMSC15_getMSC15DeviceType (
            int handle,
            int * type )
```

Ruft den Geräte-Typ des MSC15 & CSS Gerätevariante ab.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *type* | Integer Wert; gibt den Gerätetyp an:<br>• 0: nicht definiert<br>• 1: MSC15<br>• 2: MSC15-W<br>• 3: LVMH-spectralux100<br>• 4: CSS-45<br>• 5: CSS-45-WT<br>• 6: CSS-D<br>• 7: CSS-45-HI<br>• 8: MSC15-Bili |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.10 GOMDMSC15_getDetectorType()

```
int __stdcall GOMDMSC15_getDetectorType (
             int handle,
             int * type )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **CSS-D**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Prüft, ob ein externer Detektor (z.B. CSS-45, CSS-45-HI, CSS-45-WT, etc. ) am CSS-D angeschlossen ist und gibt den Gerätetyp des angeschlossenen Geräts zurück.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *type* | Integer Wert; gibt den Gerätetyp an des angelschossenen Detektors an. Die Nummer entspricht der Funktion GOMDMSC15_getMSC15DeviceType(). Z.B. type = 7 = CSS-45-HI |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.11 GOMDMSC15_getDetectorSerialNumber()

```
int __stdcall GOMDMSC15_getDetectorSerialNumber (
             int handle,
             char * value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **CSS-D**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Liefert die Seriennummer des angeschlossenen Detektors. Beispiel CSS-D + CSS-45.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Nullterminierter String; enthält nach Rücksprung die Seriennummer des angeschlossenen Detektors, Mindestgröße: 10 Bytes |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.2.3.12 GOMDMSC15_getTemperature()

```
int __stdcall GOMDMSC15_getTemperature (
            int handle,
            double * value )
```

Liest die aktuelle Leiterkartentemperatur des Geräts aus. Dies kann für relative Temperaturmessungen benutzt werden.
MSC15: Leiterkartentemperatur MSC-15 - Versions
CSS-45: Leiterkartentemperatur CSS-45 - Versons
CSS-D + CSS-45: Leiterkartentemperatur CSS-45
--> Senor liest immer die Temperatur des Sensors auf

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Double Wert, enthält die Temperatur. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

## 5.3 Methoden für Geräteeinstellungen

### Funktionen

- int __stdcall GOMDMSC15_setDisplayMode (int handle, int mode)
- int __stdcall GOMDMSC15_getDisplayMode (int handle, int ∗mode)
- int __stdcall GOMDMSC15_getAvailableDisplays (int handle, int ∗mode)
- int __stdcall GOMDMSC15_setDisplayOrientation (int handle, bool rotated)
- int __stdcall GOMDMSC15_getDisplayOrientation (int handle, bool ∗rotated)

### 5.3.1 Ausführliche Beschreibung

Geräteeinstellungen vom MSC15 & CSS Gerätevariante aufrufen und verändern.

### 5.3.2 Dokumentation der Funktionen

#### 5.3.2.1 GOMDMSC15_setDisplayMode()

```
int __stdcall GOMDMSC15_setDisplayMode (
              int handle,
              int mode )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15, MSC15-W, MSC15-Bili, CSS-45, CSS-45-WT, CSS-45-HI, CSS-D + Detector**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Funktion aktiviert/deaktiviert bestimmte Displayfunktionen im MSC15 & CSS Gerätevariante.
Das CSS-45 besitzt kein Display, allerdings werden die Displayeinstellungen im Verbund mit einer Anzeigeeinheit (CSS-D) im CSS-45 gespeichert. Das CSS-D dient nur als reines Anzeigeelement.

Beachten Sie: Nicht alle Displayfunktionen stehen bei jeder Gerätevariante zur Verfügung. Für die verfügbaren Displayfunktionen rufen Sie die Funktion GOMDMSC15_getAvailableDisplays() auf.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |

**Parameter**

| | | |
|---|---|---|
| in | *mode* | Integer Wert, wird bit-weise interpretiert:<br>• Bit 0: Grafik / E / CT (nur MSC15)<br>• Bit 1: CRI (nur MSC15)<br>• Bit 2: E / CT (nur MSC15)<br>• Bit 3: ES / EP (nur MSC15)<br>• Bit 4: Grafik / PAR (nur MSC15)<br>• Bit 5: Grafik / Bilirubin (veraltet)<br>• Bit 6: Melanopic / Brightness Factor (veraltet)<br>• Bit 7: Grafik / E / CT Irradiance (nur MSC15)<br>• Bit 8: frei<br>• Bit 9: Radiant Power für bis zu 8 Wellenlängenbereiche (nur MSC15-W)<br>• Bit 10: Irradiance für bis zu 8 Wellenlängenbereiche (nur MSC15-W)<br>• Bit 11: CIE1931 Farbdreieck (nur MSC15)<br>• Bit 12: Grafik / Radiant Power / Irradiance (nur MSC15-W)<br>• Bit 13: Grafik / Irradiance (nur MSC15-W)<br>• Bit 14: Meter Only (nur MSC15-W)<br>• Bit 15: Grafik / Bilirubin IEC (nur MSC15)<br>• Bit 16: Grafik / Bilirubin AAP (nur MSC15)<br>• Bit 17: Melanopic (nur MSC15)<br>• Bit 18: TM30-15 (nur MSC15)<br>• Bit 19: Grafik / Bilirubin BLUE LED (nur MSC15-Bili)<br>• Bit 20: Grafik / Bilirubin AAP(2004) (nur MSC15-Bili)<br>• Bit 21: Bilirubin IEC (nur MSC15-Bili)<br>• Bit 22: Bilirubin AAP(2011) (nur MSC15-Bili)<br>• Bit 23: Bilirubin BLUE LED (nur MSC15-Bili)<br>• Bit 24: Bilirubin AAP(2004) (nur MSC15-Bili) |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.3.2.2 GOMDMSC15_getDisplayMode()

```
int __stdcall GOMDMSC15_getDisplayMode (
            int handle,
            int * mode )
```

/note MSC15, MSC15-W, MSC15-Bili, CSS-45, CSS-45-WT, CSS-45-HI, CSS-D + Detector

Diese Funktion liest die Displayfunktionen aus welche im Gerät akiviert sind.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *mode* | Pointer auf Integer Wert, wird bit-weise interpretiert, siehe setDisplayMode(). |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.3.2.3 GOMDMSC15_getAvailableDisplays()

```
int __stdcall GOMDMSC15_getAvailableDisplays (
            int handle,
            int * mode )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15, MSC15-W, MSC15-Bili, CSS-45, CSS-45-WT, CSS-45-HI, CSS-D + Detector**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Funktion liest die Displayfunktionen aus die im Gerät verfügbar sind.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *mode* | Pointer auf Integer Wert, wird bit-weise interpretiert, siehe setDisplayMode(). |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.3.2.4 GOMDMSC15_setDisplayOrientation()

```
int __stdcall GOMDMSC15_setDisplayOrientation (
            int handle,
            bool rotated )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Funktion legt die Orientierung des Gerätedisplays fest.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *rotated* | Boolean Wert, ob Display um 180° gedreht wird.<br>• true: Display wird um 180° gedreht<br>• false: Display wird nicht gedreht |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.3.2.5 GOMDMSC15_getDisplayOrientation()

```
int __stdcall GOMDMSC15_getDisplayOrientation (
             int handle,
             bool * rotated )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Funktion liest die Orientierung des Geräte-Displays aus.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *rotated* | Pointer auf Boolean Wert:<br>• true: Display ist um 180° gedreht bzw. das Display steht auf dem Kopf<br>• false: Display ist nicht gedreht |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

## 5.4 Messmethoden

### Funktionen

- int __stdcall GOMDMSC15_measure (int handle)
- int __stdcall GOMDMSC15_measureWithManualIntegrationTime (int handle, int index)
- int __stdcall GOMDMSC15_measureDarkOffset (int handle)
- int __stdcall GOMDMSC15_setDynamicDarkMode (int handle, int mode)
- int __stdcall GOMDMSC15_isOffsetInvalid (int handle, bool *value)
- int __stdcall GOMDMSC15_getOffsetTime (int handle, int *timeInMs)
- int __stdcall GOMDMSC15_measurePulse (int handle, double pulseTime, double samplingRate)
- int __stdcall GOMDMSC15_dequeuePulse (int handle, double *timer, double *value)
- int __stdcall GOMDMSC15_abortPulse (int handle)

### 5.4.1 Ausführliche Beschreibung

Methoden um eine Messung durchzuführen.

### 5.4.2 C++ Aufrufbeispiel

Messung mit MSC15 & CSS Gerätevariante durchführen.

```
GOMDMSC15_getHandle(NULL, &handle);     //Initialisierung des device
GOMDMSC15_measure(handle);              //Messung durchführen
        //nach measure() könnten z.B. die Farbwerte aus dem Gerät ausgelesen werden
        //Messwerte verarbeiten
GOMDMSC15_releaseHandle(handle);        //Reinitialisierung, gibt Recourcen wieder frei
```

### 5.4.3 Dokumentation der Funktionen

#### 5.4.3.1 GOMDMSC15_measure()

```
int __stdcall GOMDMSC15_measure (
              int handle )
```

Diese Methode stößt die Messung an. Die Messwerte werden danach im Gerät hinterlegt und können u.a. mit den Funktionen im Kapitel

- Spektrale Messwertmethoden oder
- Methoden zum Auslesen der Farbwerte

abgefragt werden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.2 GOMDMSC15_measureWithManualIntegrationTime()

```
int __stdcall GOMDMSC15_measureWithManualIntegrationTime (
              int handle,
              int index )
```

Diese Methode stößt die Messung mit manueller Integrationszeit an. Die Messwerte werden danach im Gerät hinterlegt und können u.a. mit den Funktionen im Kapitel

- Spektrale Messwertmethoden oder
- Methoden zum Auslesen der Farbwerte

abgefragt werden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Integer Wert mit Integrationszeit verknüpft:<br>• 0: Integrationszeit: 0.000012 s<br>• 1: Integrationszeit: 0.00002 s<br>• 2: Integrationszeit: 0.00004 s<br>• 3: Integrationszeit: 0.00008 s<br>• 4: Integrationszeit: 0.00016 s<br>• 5: Integrationszeit: 0.00032 s<br>• 6: Integrationszeit: 0.00064 s<br>• 7: Integrationszeit: 0.00125 s<br>• 8: Integrationszeit: 0.0025 s<br>• 9: Integrationszeit: 0.005 s<br>• 10: Integrationszeit: 0.01 s<br>• 11: Integrationszeit: 0.02 s<br>• 12: Integrationszeit: 0.04 s<br>• 13: Integrationszeit: 0.08 s<br>• 14: Integrationszeit: 0.16 s<br>• 15: Integrationszeit: 0.32 s<br>• 16: Integrationszeit: 0.64 s<br>• 17: Integrationszeit: 1.28 s<br>• 18: Integrationszeit: 2.56 s |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.3 GOMDMSC15_measureDarkOffset()

```
int __stdcall GOMDMSC15_measureDarkOffset (
              int handle )
```

Der Dunkel Offset wird gemessen. Bei einem MSC-15 muss der Shutter dazu am Gerät manuell geschlossen und nach der Messung wieder geöffnet werden. Bei einem CSS-45 wird der Sutter automatisch geschlossen und geöffnet. Der gemessene Offset wird automatisch von den (Hell-)Messungen abgezogen.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.4 GOMDMSC15_setDynamicDarkMode()

```
int __stdcall GOMDMSC15_setDynamicDarkMode (
              int handle,
              int mode )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **CSS-45, CSS-45-HI, CSS-45-WT**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese methode steht nur für CSS-Geräte zur verfügung.
Wenn aktiviert(mode = 1) wird nach jeder Messung eine Dunkelmessung mit der entsprechenden Integrationszeit durchgeführt und für die Berechnung verwendet.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *mode* | Integer Wert:<br>• 0: dynamischer Dunkelmodus wird deaktiviert<br>• 1: dynamischer Dunkelmodus wird aktiviert |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.5 GOMDMSC15_isOffsetInvalid()

```
int __stdcall GOMDMSC15_isOffsetInvalid (
            int handle,
            bool * value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **CSS-45, CSS-45-HI, CSS-45-WT**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Die Methode gibt an, ob der Dunkelstrom-Abzug im Gerät noch gültig ist. Wenn nicht, muss er mit der Methode GOMDMSC15_measureDarkOffset() neu gemessen werden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Boolean Wert:<br>• true: Dunkelabzug ist ungültig<br>• false: Dunkelabzug ist nicht ungültig |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.6 GOMDMSC15_getOffsetTime()

```
int __stdcall GOMDMSC15_getOffsetTime (
            int handle,
            int * timeInMs )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **CSS-45, CSS-45-HI, CSS-45-WT**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Gibt die Zeit an, die benötigt wird um eine Offsetmessung durchzuführen.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *timeInMs* | Zeit für Offset-Messung in ms. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.7 GOMDMSC15_measurePulse()

```
int __stdcall GOMDMSC15_measurePulse (
            int handle,
            double pulseTime,
            double samplingRate )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Methode stößt eine Pulsmessung an. Eine Pulsmessung besteht aud sehr schnellen Einzelmessungen, mit der ein Pulsverlauf aufgezeichnet wird. Die Funktion kommt erst zurück, wenn die gesamte Messung abgeschlossen ist.
Mit der Funktion dequeuePulse() können in einem separaten Thread die Messzeiten und radiometrischen Werte der einzelnen Messungen ausgelesen werden.
Nach der Pulsmessung sind die Messwerte für die Einzelmessung mit dem größten radiometrischen Integral im SDK hinterlegt und können mit den Funktionen im Kapitel „Auslesen der Messwerte“ abgefragt werden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *pulseTime* | Die Dauer der Pulsmessung in Sekunden |
| in | *samplingRate* | Die Abtastrate in Hz. Sie legt fest, wie schnell hintereinander die Einzelmessungen stattfinden sollen. Wenn nicht genug Signal vorhanden ist, kann es sein, dass die eingestellte Abtastrate nicht erreicht wird, da die Dauer der Einzelmessungen zu lange ist. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.8 GOMDMSC15_dequeuePulse()

```
int __stdcall GOMDMSC15_dequeuePulse (
             int handle,
             double * timer,
             double * value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Funktion liest während oder nach einer Pulsmessung die Messzeiten und radiometrischen Messwerte aus. Sie ist threadsicher und kann während der Pulsmessung in einem separaten Thread aufgerufen werden, um die Werte der aktuellen Pulsmessung auszulesen.
Sie sollte in einer Schleife solange aufgerufen werden, bis alle Messwerte abgeholt wurden. Liefert die Funktion den Rückgabewert -25030 (Fehler), bedeutet dies, dass derzeit keine Messwerte vorhanden sind.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *timer* | Messzeit der Einzelmessung. |
| out | *value* | radiometrischer Wert der Einzelmessung. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.4.3.9 GOMDMSC15_abortPulse()

```
int __stdcall GOMDMSC15_abortPulse (
             int handle )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Funktion bricht die aktuelle Pulsmessung ab. Sie muss in einem separaten Thread aufgerufen werden und führt dazu, dass die Funktion GOMDMSC15_measurePulse() direkt beendet wird.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |

#### Rückgabe

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

## 5.5 Spektrale Messwertmethoden

### Funktionen

- int __stdcall GOMDMSC15_getLastIntegrationTime (int handle, double ∗timeInS)
- int __stdcall GOMDMSC15_getWLBorders (int handle, double ∗startWL, double ∗endWL)
- int __stdcall GOMDMSC15_getWLMapping (int handle, double ∗wavelengths)
- int __stdcall GOMDMSC15_getSpectralDataByPixel (int handle, double ∗spectrum)
- int __stdcall GOMDMSC15_getSpectralData (int handle, double startWL, double deltaWL, int nrOfSteps, double ∗spectrum)
- int __stdcall GOMDMSC15_getPeakWL (int handle, double ∗value)
- int __stdcall GOMDMSC15_getCentreWL (int handle, double ∗value)
- int __stdcall GOMDMSC15_getCentroidWL (int handle, double ∗value)
- int __stdcall GOMDMSC15_getFWHM (int handle, double ∗value)

### 5.5.1 Ausführliche Beschreibung

Methoden um spektrale Messwerte aufzurufen.

### 5.5.2 C++ Aufrufbeispiel

Spektrum in pixelbasierten Schritten (288 Pixel) aus dem Gerät auslesen.

```
int handle;
GOMDMSC15_getHandle(NULL, &handle);             //Initialisierung des device
double spectrum[288];                           //Array für die 288 Pixel
GOMDMSC15_getSpectralData(handle, &spectrum);   //Spektrum auslesen
        //Daten von ßpectrum" verarbeiten
GOMDMSC15_releaseHandle(handle);                //Reinitialisierung, gibt Recourcen wieder frei
```

### 5.5.3 Dokumentation der Funktionen

#### 5.5.3.1 GOMDMSC15_getLastIntegrationTime()

```
int __stdcall GOMDMSC15_getLastIntegrationTime (
            int handle,
            double * timeInS )
```

Diese Funktion gibt die Messdauer (Integrationszeit) der letzten Messung aus. Letzte Messung kann u.a. sein interne Mssung des Geräts bei der Initialisierung, GOMDMSC15_measureDarkOffset(), GOMDMSC15_measure(), ... . D.h. es ist ratsam diese Methode direkt nach der ausgeführten Messung aufzurufen.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *timeInS* | Pointer auf Doublewert: Integrationszeit in Sekunden |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.2 GOMDMSC15_getWLBorders()

```
int __stdcall GOMDMSC15_getWLBorders (
            int handle,
            double * startWL,
            double * endWL )
```

Diese Funktion ruft die Wellenlängenbegrenzung des Geräts ab.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *startWL* | Pointer auf Doublewert: Wellenlängen Start. |
| out | *endWL* | Pointer auf Doublewert: Wellenlängen Ende. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.3 GOMDMSC15_getWLMapping()

```
int __stdcall GOMDMSC15_getWLMapping (
            int handle,
            double * wavelengths )
```

Diese Funktion ruft die Wellenlängenstützpunkte für die 288 Pixel des Gerätes ab.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *wavelengths* | Pointer auf Array von Double-Werten; Array mit den Wellenlängenstützpunkten. Das Array muss mit einer Größe von 288 vorinitialisiert sein. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.4 GOMDMSC15_getSpectralDataByPixel()

```
int __stdcall GOMDMSC15_getSpectralDataByPixel (
             int handle,
             double * spectrum )
```

Ermittelt mit den gemessenen Daten und dem Kalibrierfaktoren das Spektrum und gibt dieses in pixelbasierten Schritten aus. Mit getWLMapping können die Wellenlängen abgerufen werden, die zu den 288 Pixeln gehören.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *spectrum* | Pointer auf das erste Element eines Double Array, enthält die berechneten Messwerte. Die Größe des Arrays muss auf 288 festgelegt sein. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.5 GOMDMSC15_getSpectralData()

```
int __stdcall GOMDMSC15_getSpectralData (
             int handle,
             double startWL,
             double deltaWL,
             int nrOfSteps,
             double * spectrum )
```

Ermittelt mit den gemessenen Daten und dem Kalibrierfaktoren das Spektrum und gibt dieses im angegebenen Bereich(Startwellenlänge, Schrittweite und Anzahl der Werte) aus. Das Spektrum wird dabei auf den gewünschten Bereich interpoliert. Mit getWLBorders können die maximal möglichen Grenzen für das Spektrum ausgelesen werden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *startWL* | Startwellenlänge |
| in | *deltaWL* | Schrittweite der Wellenlänge zwischen den einzelen Werten. |
| in | *nrOfSteps* | Anzahl an Ausgabewerten. Dadurch wird auch automatisch die Endwellenlänge festgelegt. |
| out | *spectrum* | Pointer auf das erste Element eines Double Array, enthält die berechneten Messwerte. Die Größe des Arrays wird mit nrOfSteps definiert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.6 GOMDMSC15_getPeakWL()

```
int __stdcall GOMDMSC15_getPeakWL (
             int handle,
             double * value )
```

Liefert die Wellenlänge bei der das Spektrum maximal ist.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält die Wellenlänge in [nm]. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.7 GOMDMSC15_getCentreWL()

```
int __stdcall GOMDMSC15_getCentreWL (
             int handle,
             double * value )
```

Liefert die Zentrumswellenläge.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält den Mittelpunkt der Zentrumswellenlänge in [nm]. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.8 GOMDMSC15_getCentroidWL()

```
int __stdcall GOMDMSC15_getCentroidWL (
             int handle,
             double * value )
```

Liefert die Schwerpunktwellenlänge. Die Schwerpunktwellenlänge ist ein Maß um ein Spektrum zu charakterisieren. Sie gibt an, wo sich der "Mittelpunkt" des Spektrums befindet.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält die Schwerpunktwellenlänge in [nm]. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.5.3.9 GOMDMSC15_getFWHM()

```
int __stdcall GOMDMSC15_getFWHM (
            int handle,
            double * value )
```

Liefert die Halbwertsbreite (FDHM = full width at half maximum) des gemessenen Spektrums.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält die Halbwertsbreite in [nm]. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

# 5.6 Integrale Messwertmethoden

## Funktionen

- int __stdcall GOMDMSC15_getPhotopic (int handle, double ∗value)
- int __stdcall GOMDMSC15_getScotopic (int handle, double ∗value)
- int __stdcall GOMDMSC15_getSPRatio (int handle, double ∗value)
- int __stdcall GOMDMSC15_getRadiometricUnit (int handle, int ∗unit)
- int __stdcall GOMDMSC15_getRadiometricValue (int handle, double startWL, double endWL, double ∗value)
- int __stdcall GOMDMSC15_getPAR (int handle, double ∗value)
- int __stdcall GOMDMSC15_getBilirubinIEC (int handle, double ∗value)
- int __stdcall GOMDMSC15_getBilirubinAAP (int handle, double ∗value)
- int __stdcall GOMDMSC15_getBilirubinAAP2004 (int handle, double ∗value)
- int __stdcall GOMDMSC15_getBilirubinNatus (int handle, double ∗value)
- int __stdcall GOMDMSC15_getMelanopic (int handle, double ∗Ev, double ∗Eemel, double ∗EvmelD65)

### 5.6.1 Ausführliche Beschreibung

Methoden um integrale Messwerte aufzurufen.

### 5.6.2 C++ Aufrufbeispiel

Den Wert PAR [µmol/m2/s] aus dem Gerät auslesen.

```
int handle;
double value;
GOMDMSC15_getHandle(NULL, &handle);     //Initialisierung des device
GOMDMSC15_measure();                    //Messung durchführen
GOMDMSC15_ getPAR(handle, &value);      //Messwert aufrufen
        //Daten von "value" verarbeiten
GOMDMSC15_releaseHandle(handle);        //Reinitialisierung, gibt Recourcen wieder frei
```

### 5.6.3 Dokumentation der Funktionen

#### 5.6.3.1 GOMDMSC15_getPhotopic()

```
int __stdcall GOMDMSC15_getPhotopic (
            int handle,
            double * value )
```

Liefert den photopischen Wert des zuletzt gemessenen Spektrums.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält nach Rücksprung den ermittelten photopischen Wert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.2 GOMDMSC15_getScotopic()

```
int __stdcall GOMDMSC15_getScotopic (
            int handle,
            double * value )
```

Liefert den scotopischen Wert des zuletzt gemessenen Spektrums.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $> 0$ zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält nach Rücksprung den ermittelten photopischen Wert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.3 GOMDMSC15_getSPRatio()

```
int __stdcall GOMDMSC15_getSPRatio (
            int handle,
            double * value )
```

Liefert das Verhältnis aus scotopischem und photopischem Wert.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $> 0$ zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält nach Rücksprung das scotopisch/photopische Verhältnis. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.4 GOMDMSC15_getRadiometricUnit()

```
int __stdcall GOMDMSC15_getRadiometricUnit (
```

```
            int handle,
            int * unit )
```

Diese Funktion gibt die Einheit des Radiometrischen Wertes an.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *unit* | Pointer auf Integer Wert:<br>• 0: 'W' (MSC15-W)<br>• 1: 'W/m2' |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.5 GOMDMSC15_getRadiometricValue()

```
int __stdcall GOMDMSC15_getRadiometricValue (
            int handle,
            double startWL,
            double endWL,
            double * value )
```

Diese Funktion gibt den Wert des Radiometrischen Integrals an.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *startWL* | Start-Wert, des Bereichs in dem das Integral berechnet werden soll. |
| in | *endWL* | End-Wert, des Bereichs in dem das Integral berechnet werden soll. |
| out | *value* | Pointer auf Double Wert: Radiometrisches Integral im angegebenen Wellenlängenbereich. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.6 GOMDMSC15_getPAR()

```
int __stdcall GOMDMSC15_getPAR (
            int handle,
            double * value )
```

Diese Methode liefert den Wert von PAR in µmol/m2/s.

**Siehe auch**

https://www.gigahertz-optik.de/de-de/service-und-support/informationsportal/par/

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält den gemessenen Wert von PAR. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.7 GOMDMSC15_getBilirubinIEC()

```
int __stdcall GOMDMSC15_getBilirubinIEC (
            int handle,
            double * value )
```

Diese Methode liefert den Wert von Bilirubin nach IEC 60601-2-50.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält Wert von Bilirubin nach IEC 60601-2-50. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.8 GOMDMSC15_getBilirubinAAP()

```
int __stdcall GOMDMSC15_getBilirubinAAP (
            int handle,
            double * value )
```

Diese Methode liefert den Wert von Bilirubin nach der AAP(American Academy of Pediatrics) von 2011.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält Wert von Bilirubin nach AAP 2004. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.9 GOMDMSC15_getBilirubinAAP2004()

```
int __stdcall GOMDMSC15_getBilirubinAAP2004 (
            int handle,
            double * value )
```

Diese Methode liefert den Wert von Bilirubin nach der AAP(American Academy of Pediatrics) von 2004.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält Wert von Bilirubin nach AAP 2004. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.10 GOMDMSC15_getBilirubinNatus()

```
int __stdcall GOMDMSC15_getBilirubinNatus (
            int handle,
            double * value )
```

Diese Methode liefert den Wert von Bilirubin gefordert von Natus einer blaunen LED.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält Wert von Bilirubin einer blaunen LED. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.6.3.11 GOMDMSC15_getMelanopic()

```
int __stdcall GOMDMSC15_getMelanopic (
            int handle,
```

```
double * Ev,
double * Eemel,
double * EvmelD65 )
```

Diese Methode liefert die melanopischen Werte zurück.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *Ev* | Pointer auf Double Wert, Beleuchtungsstärke. |
| out | *Eemel* | Pointer auf Double Wert, Melanopische Bestrahlungsstärke. |
| out | *EvmelD65* | Pointer auf Double Wert, Tageslicht-äquivalente(D65) melanopische Beleuchtungsstärke. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

# 5.7 Methoden zum Auslesen der Farbwerte

## Funktionen

- int __stdcall GOMDMSC15_getColor (int handle, double ∗XValue, double ∗YValue, double ∗ZValue)
- int __stdcall GOMDMSC15_getColorCIE1931 (int handle, double ∗xValue, double ∗yValue)
- int __stdcall GOMDMSC15_getColorCIE1976 (int handle, double ∗uValue, double ∗vValue)
- int __stdcall GOMDMSC15_getCRI (int handle, double ∗values)
- int __stdcall GOMDMSC15_getCCT (int handle, double ∗value)
- int __stdcall GOMDMSC15_getDomWL (int handle, double ∗value)
- int __stdcall GOMDMSC15_getCompWL (int handle, double ∗value)
- int __stdcall GOMDMSC15_getDeltauv (int handle, double ∗value)
- int __stdcall GOMDMSC15_getPurity (int handle, double ∗value)
- int __stdcall GOMDMSC15_getTM3018 (int handle, double ∗Rf, double ∗Rg)
- int __stdcall GOMDMSC15_getTM3018RfByHue (int handle, double ∗values)

### 5.7.1 Ausführliche Beschreibung

Mit diesen Mehoden können Farbwerte aus dem Gerät ausgelesen werden.

### 5.7.2 C++ Aufrufbeispiel

Auslesen der Farbkoordinaten u‘ und v‘ des CIE1976 Farbraums.

```
int handle;
GOMDMSC15_getHandle(NULL, &handle);             //Initialisierung des device
GOMDMSC15_measure();                            //Messung durchführen
double u, v;
GOMDMSC15_getColorCIE1976(handle, &u, &v);      //Nach Return enthält u, v die Farbkoordinaten
        //Daten von "value" verarbeiten
GOMDMSC15_releaseHandle(handle);                //Reinitialisierung, gibt Recourcen wieder frei
```

### 5.7.3 Dokumentation der Funktionen

#### 5.7.3.1 GOMDMSC15_getColor()

```
int __stdcall GOMDMSC15_getColor (
            int handle,
            double * XValue,
            double * YValue,
            double * ZValue )
```

Diese Methode liefert alle berechneten Farbwerte auf Basis einer spektralen Messung. Die Spektralmessung und die Farbberechnung müssen vor der Messung aktiviert worden sein.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *XValue* | Pointer auf Double Wert, enthält “groß” X. |
| out | *YValue* | Pointer auf Double Wert, enthält “groß” Y. |
| out | *ZValue* | Pointer auf Double Wert, enthält “groß” Z. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.2 GOMDMSC15_getColorCIE1931()

```
int __stdcall GOMDMSC15_getColorCIE1931 (
            int handle,
            double * xValue,
            double * yValue )
```

Diese Methode liefert die x und y Farbkoordinaten des CIE1931 Farbraums.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *xValue* | Pointer auf Double Wert, enthält x entsprechend dem CIE1931 Farbraum. |
| out | *yValue* | Pointer auf Double Wert, enthält y entsprechend dem CIE1931 Farbraum. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.3 GOMDMSC15_getColorCIE1976()

```
int __stdcall GOMDMSC15_getColorCIE1976 (
            int handle,
            double * uValue,
            double * vValue )
```

Diese Methode liefert die u‘ und v‘ Farbkoordinaten des CIE1976 Farbraums.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *uValue* | Pointer auf Double Wert, enthält u‘ entsprechend dem CIE1976 Farbraum |
| out | *vValue* | Pointer auf Double Wert, enthält v‘ entsprechend dem CIE1976 Farbraum |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.4 GOMDMSC15_getCRI()

```
int __stdcall GOMDMSC15_getCRI (
            int handle,
            double * values )
```

Unter Farbwiedergabeindex (englisch Colour Rendering Index, CRI) versteht man eine photometrische Größe, mit der sich die Qualität der Farbwiedergabe von Lichtquellen gleicher korrelierter Farbtemperatur beschreiben lässt. Als Referenz zur Beurteilung der Wiedergabequalität dient bis zu einer Farbtemperatur von 5000 K das Licht, das von einem schwarzen Strahler der entsprechenden Farbtemperatur abgegeben wird. Über 5000 K wird gegenüber einer tageslichtähnlichen Spektralverteilung referenziert. Beispielsweise wird für die Berechnung der Farbwiedergabe einer Haushaltsglühlampe (die selbst in guter Näherung ein schwarzer Strahler ist) das Spektrum eines schwarzen Strahlers mit einer Temperatur von 2700 K als Referenz verwendet, für eine Leuchtstofflampe mit der Lichtfarbe 865 (865 für einen Farbwiedergabeindex von mehr als 80, 865 für eine Farbtemperatur von 6500 K) dagegen das Tageslichtspektrum der Normlichtart D65. Der Farbwiedergabeindex ist seiner Definition nach ein spezieller Metamerieindex. Zur Berechnung des Farbwiedergabeindex sind 14 Testfarben mit einem genormten Remissionsverlauf definiert. Die Abweichung der Sekundärspektren zwischen Referenz- und Testspektrum dient als Maßzahl für die 14 speziellen Farbwiedergabeindizes. Zur Berechnung des allgemeinen Farbwiedergabeindex Ra werden allerdings nur die ersten acht Testfarben herangezogen. Die 14 Testfarben sind durch DIN 6169 ausgewählt. Dabei kann der Farbwiedergabeindex Ri zur Farbe i ermittelt werden. Ein rechnerischer Wert aus den Farben #1 bis #8 wird mit Ra bezeichnet. Da bei der Festlegung des Farbwiedergabeindex in den 1930er Jahren die Referenzlichtquellen mit 100, die damals gängigen Leuchtstofflampen (gewissermaßen willkürlich) mit 50 festgesetzt wurden und der Farbwiedergabeindex keinesfalls ein prozentualer Wert ist, sind auch negative Farbwiedergabeindizes möglich.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *values* | Pointer auf das erste Element eines Double Array. Dieses enthält nach Rückgabe die berechneten CRI Werte. Die Größe des Arrays muss mit 16 vorinitialisiert sein. values[0] enthält dann den Ra Wert. Die Einträge values[1] - values[15] enthalten die Werte R1 - R15. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.5 GOMDMSC15_getCCT()

```
int __stdcall GOMDMSC15_getCCT (
            int handle,
            double * value )
```

Liefert die Farbtemperatur(CCT) der letzten Messung.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält nach Rücksprung den ermittelten CCT Wert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.6 GOMDMSC15_getDomWL()

```
int __stdcall GOMDMSC15_getDomWL (
             int handle,
             double * value )
```

Gibt den Wert der dominanten Wellenlänge der letzten Messung zurück

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält nach Rücksprung den ermittelten Wert der dominanten Wellenlänge |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.7 GOMDMSC15_getCompWL()

```
int __stdcall GOMDMSC15_getCompWL (
             int handle,
             double * value )
```

Gibt den Wert der komplementären Wellenlänge der letzten Messung zurück

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält nach Rücksprung den ermittelten Wert der komplementären Wellenlänge |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.8 GOMDMSC15_getDeltauv()

```
int __stdcall GOMDMSC15_getDeltauv (
             int handle,
             double * value )
```

Liefert den bei der letzten Messung tatsächlich vorhandenen delta uv Wert.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $> 0$ zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält nach Rücksprung den ermittelten delta uv Wert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.9 GOMDMSC15_getPurity()

```
int __stdcall GOMDMSC15_getPurity (
             int handle,
             double * value )
```

Liefert Farbreinheit der Letzten Messung.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $> 0$ zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält die Farbreinheit. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.10 GOMDMSC15_getTM3018()

```
int __stdcall GOMDMSC15_getTM3018 (
             int handle,
             double * Rf,
             double * Rg )
```

**TM-30-18** ist ein Farbwiedergabeindex der durch die IES veröffentlicht wurde. Im Gegensatz zum CRI beruht er auf 99 unterschiedlichen Testfarben, um eine möglichst genaue und an alle Lichtarten angepasstes Bewertungssystem zu erhalten. Die wichtigsten Kennziffern in TM-30-18 sind die Werte Rf(Fidelity-Index) und Rg(Gamut-Index). Rf ist vom Prinzip vergleichbar mit dem Ra-Wert des CRI. Der Rg-Wert gibt zusätzlich die Sättigung der Farben unter dem gemessenen Licht an.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *Rf* | Pointer auf Double Wert: Fidelity-Index |
| out | *Rg* | Pointer auf Double Wert: Gammut-Index |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.7.3.11 GOMDMSC15_getTM3018RfByHue()

```
int __stdcall GOMDMSC15_getTM3018RfByHue (
            int handle,
            double * values )
```

Beim CRI gibt es die Werte R1-R15. Um beim TM-30-18 eine Aussage über die Farbqualität für einzelnen Farben geben zu können gibt es hier die Aufspaltung des Rf in 16 gemittelte Farbwiedergabe Werte. Der Rf[by Hue].

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *values* | Pointer auf das erste Element eines Double Array. Dieses enthält die berechneten Rf-Werte nach Farbe. Die Größe des Arrays muss mit 16 vorinizialisiert sein. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

# 5.8 Laden von gespeicherten Messungen

## Funktionen

- int __stdcall GOMDMSC15_loggerReadDate (int handle, int index, char ∗date)
- int __stdcall GOMDMSC15_loggerLoadMeasurement (int handle, int index)
- int __stdcall GOMDMSC15_loggerDeleteMeasurement (int handle, int index)
- int __stdcall GOMDMSC15_loggerSetRTC (int handle, char ∗date)
- int __stdcall GOMDMSC15_loggerGetRTC (int handle, char ∗date)

## 5.8.1 Ausführliche Beschreibung

Methoden um gespeicherte Messung aus dem Geräte Logger auszulesen und zu löschen.

Das speichern der Messungen im Gerätelogger ist nur im Handmodus möglich.

## 5.8.2 Dokumentation der Funktionen

### 5.8.2.1 GOMDMSC15_loggerReadDate()

```
int __stdcall GOMDMSC15_loggerReadDate (
              int handle,
              int index,
              char * date )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15, MSC15-Bili, CSS-45, CSS-45-HI, CSS-45-WT, CSS-45 + CSS-D**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Methode liest das Datum einer Messung aus dem internen Datanlogger aus.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Index der Messung im internen Datenlogger (0-9) |
| out | *date* | Nullterminierter String; enthält nach Rücksprung die das Datum und die Uhrzeit(TTMMJJhhmmss) |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.8.2.2 GOMDMSC15_loggerLoadMeasurement()

```
int __stdcall GOMDMSC15_loggerLoadMeasurement (
              int handle,
              int index )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15, MSC15-Bili, CSS-45, CSS-45-HI, CSS-45-WT, CSS-45 + CSS-D**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Methode ladet eine Messung aus dem internen Datanlogger. Auf die Daten kann dann mit den allegmeinen Methoden zum Auslesen von Messdaten zugegriffen werden. Die aktuellen Messwerte werden beim Aufruf dieser Methode überschrieben.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Index der Messung im internen Datenlogger (0-9) |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.8.2.3 GOMDMSC15_loggerDeleteMeasurement()

```
int __stdcall GOMDMSC15_loggerDeleteMeasurement (
              int handle,
              int index )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15, MSC15-Bili, CSS-45, CSS-45-HI, CSS-45-WT, CSS-45 + CSS-D**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Methode löscht die Messung mit dem übergebenem Index aus dem internen Datenlogger.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Index der Messung im internen Datenlogger (0-9) |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.8.2.4 GOMDMSC15_loggerSetRTC()

```
int __stdcall GOMDMSC15_loggerSetRTC (
              int handle,
              char * date )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15, MSC15-Bili, CSS-D**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Methode setzt die interne Echtzeit-Uhr. Sie wird für den Geräte-Logger verwendet.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *date* | Nullterminierter String; enthält das Datum und die Uhrzeit(TTMMJJhhmmss) |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.8.2.5 GOMDMSC15_loggerGetRTC()

```
int __stdcall GOMDMSC15_loggerGetRTC (
              int handle,
              char * date )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15, MSC15-Bili, CSS-D**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Diese Methode liest die interne Echtzeit-Uhr aus. Sie wird für den Geräte-Logger verwendet.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *date* | Nullterminierter String; enthält nach Rücksprung das Datum und die Uhrzeit(TTMMJJhhmmss) |

#### Rückgabe

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

## 5.9 Methoden für MSC15-W-Geräte

### Funktionen

- int __stdcall GOMDMSC15_setUserWLRangeBorders (int handle, int userRange, double startWL, double endWL)
- int __stdcall GOMDMSC15_setUserWLRangeBoarders (int handle, int userRange, double startWL, double endWL)
- int __stdcall GOMDMSC15_setUserWLRangeBordersToFactory (int handle)
- int __stdcall GOMDMSC15_getUserWLRangeBorders (int handle, int userRange, double *startWL, double *endWL)
- int __stdcall GOMDMSC15_getUserWLRangeBoarders (int handle, int userRange, double *startWL, double *endWL)
- int __stdcall GOMDMSC15_getSelectedUserWLRange (int handle, int *userRange, bool *locked)
- int __stdcall GOMDMSC15_setSelectedUserWLRange (int handle, int userRange, bool locked)
- int __stdcall GOMDMSC15_getUserWLRangeActive (int handle, int userRange, bool *active)
- int __stdcall GOMDMSC15_getUserWLRangeModifiable (int handle, int userRange, bool *modifiable)
- int __stdcall GOMDMSC15_getUserWLRangeRadiometric (int handle, int userRange, double *value)
- int __stdcall GOMDMSC15_setAppertureSize (int handle, int index, double size)
- int __stdcall GOMDMSC15_getAppertureSize (int handle, int index, double *size)
- int __stdcall GOMDMSC15_getAppertureSizeBorders (int handle, double *minSize, double *maxSize)
- int __stdcall GOMDMSC15_setAppertureIndex (int handle, int index)
- int __stdcall GOMDMSC15_getAppertureIndex (int handle, int *index)
- int __stdcall GOMDMSC15_setUserCorrectionFactor (int handle, double value)
- int __stdcall GOMDMSC15_getUserCorrectionFactor (int handle, double *value)
- int __stdcall GOMDMSC15_setUserCorrectionFactorLocked (int handle, bool value)
- int __stdcall GOMDMSC15_getUserCorrectionFactorLocked (int handle, bool *value)

### 5.9.1 Ausführliche Beschreibung

Diese Methoden können nur in Kombination mit einem MSC15-W-Gerät verwendet werden.

### 5.9.2 Dokumentation der Funktionen

#### 5.9.2.1 GOMDMSC15_setUserWLRangeBorders()

```
int __stdcall GOMDMSC15_setUserWLRangeBorders (
            int handle,
            int userRange,
            double startWL,
            double endWL )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Setzt die Wellenlängengrenzen für die vom Benutzer einstellbaren Wellenlängen-Bereiche.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *userRange* | Integer Wert, der Wellenlängenbereich für den die Grenzen gesetzt werden sollen. (Es gibt die Bereiche 0-7). |
| in | *startWL* | Double Wert, Startwellenlänge für den Bereich. |
| in | *endWL* | Double Wert, Ende der Wellenlängenbegrenzung. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.2 GOMDMSC15_setUserWLRangeBoarders()

```
int __stdcall GOMDMSC15_setUserWLRangeBoarders (
             int handle,
             int userRange,
             double startWL,
             double endWL )
```

#### 5.9.2.3 GOMDMSC15_setUserWLRangeBordersToFactory()

```
int __stdcall GOMDMSC15_setUserWLRangeBordersToFactory (
             int handle )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Setzt die Wellenlängengrenzen für die vom Benutzer einstellbaren Wellenlängen-Bereiche auf die Werkseinstellungen zurück.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.4 GOMDMSC15_getUserWLRangeBorders()

```
int __stdcall GOMDMSC15_getUserWLRangeBorders (
             int handle,
             int userRange,
             double * startWL,
             double * endWL )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Liest die Wellenlängengrenzen für die vom Benutzer einstellbaren Wellenlängen-Bereiche aus.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *userRange* | Integer Wert, der Wellenlängenbereich für den die Grenzen ausgelesen werden sollen (Es gibt die Bereiche 0-7). |
| out | *startWL* | Pointer auf double Wert, Startwellenlänge für den Bereich. |
| out | *endWL* | Pointer auf double Wert, Ende der Wellenlängenbegrenzung. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.5 GOMDMSC15_getUserWLRangeBoarders()

```
int __stdcall GOMDMSC15_getUserWLRangeBoarders (
             int handle,
             int userRange,
             double * startWL,
             double * endWL )
```

#### 5.9.2.6 GOMDMSC15_getSelectedUserWLRange()

```
int __stdcall GOMDMSC15_getSelectedUserWLRange (
             int handle,
             int * userRange,
             bool * locked )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Ruft den Wellenlängen-Bereich ab, der aktuell für die Anzeige im Gerätedisplay verwendet wird.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *userRange* | Pointer auf integer Wert, der Wellenlängenbereich der aktuell gesetzt ist. (Es gibt die Bereiche 0-7). |
| out | *locked* | Pointer auf Boolean Wert:<br>• true: Bereich ist gesperrt<br>• false Bereich ist nicht gesperrt |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.7 GOMDMSC15_setSelectedUserWLRange()

```
int __stdcall GOMDMSC15_setSelectedUserWLRange (
            int handle,
            int userRange,
            bool locked )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Legt den Wellenlängen-Bereich fest, der für die Anzeige im Gerätedisplay verwendet wird.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert > 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *userRange* | Integer Wert, der Wellenlängenbereich der gesetzt werden soll. (Es gibt die Bereiche 0-7). |
| in | *locked* | Boolean Wert, soll dieser Bereich im Gerät gesperrt werden. Kann dann vom Benutzer nicht geändert werden. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.8 GOMDMSC15_getUserWLRangeActive()

```
int __stdcall GOMDMSC15_getUserWLRangeActive (
            int handle,
```

```
            int userRange,
            bool * active )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Ruft ab, ob ein Wellenlängenbereich aktiv ist, und somit vom Benutzer im Gerät ausgewählt werden kann. Dieser Wert wird von Gigahertz-Optik bei der Geräteauslieferung gesetzt.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *userRange* | Integer Wert, der Wellenlängenbereich der ausgelesen werden soll (Es gibt die Bereiche 0-7). |
| out | *active* | Boolean Wert:<br>• true: Bereich ist im Gerät aktiv<br>• false: Bereich ist im Gerät nicht akiv |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.9 GOMDMSC15_getUserWLRangeModifiable()

```
int __stdcall GOMDMSC15_getUserWLRangeModifiable (
            int handle,
            int userRange,
            bool * modifiable )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Ruft ab, ob ein Wellenlängenbereich mofifizierbar ist, und somit vom Benutzer im Gerät geändert werden kann. Dieser Wert wird von Gigahertz-Optik bei der Geräteauslieferung gesetzt.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *userRange* | Integer Wert, der Wellenlängenbereich der ausgelesen werden soll (Es gibt die Bereiche 0-7). |
| out | *modifiable* | Boolean Wert:<br>• true: Bereich ist im Gerät mofifizierbar<br>• false: Bereich ist im Gerät nicht mofifizierbar |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.10 GOMDMSC15_getUserWLRangeRadiometric()

```
int __stdcall GOMDMSC15_getUserWLRangeRadiometric (
            int handle,
            int userRange,
            double * value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Berechnet den Radiometrischen Wert über das Spektrum im Bereich der Wellenlängengrenzen, die mit GOMDMSC15_setUserWLRangeBoarders() festgelegt wurden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *userRange* | Integer Wert, der Wellenlängenbereich der ausgelesen werden soll (Es gibt die Bereiche 0-7). |
| out | *value* | Double Wert, enthält den radiometrischen Wert im Bereich der Wellenlängengrenzen nach Return. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.11 GOMDMSC15_setAppertureSize()

```
int __stdcall GOMDMSC15_setAppertureSize (
            int handle,
            int index,
            double size )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Legt den Aperturdurchmesser für das MSC15-W fest. Es können bis zu 5 Durchmesser im Gerät hinterlegt werden. Mit GOMDMSC15_setAppertureIndex()) kann eine Apertur ausgewählt werden. Achtung: Der Aperturdurchmesser wird bei den SDK Funktionen nicht automatisch verrechnet. Wenn Sie die Bestrahlungsstärke bestimmen möchten, müssen Sie die Berechnung selbst durchführen.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Integer Wert, der Appertur-Index, der geändert werden soll. (Es gibt die Bereiche 0-4) |
| in | *size* | Double Wert, Wert des Aperturdurchmessers. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.12 GOMDMSC15_getAppertureSize()

```
int __stdcall GOMDMSC15_getAppertureSize (
             int handle,
             int index,
             double * size )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Liest den Aperturdurchmesser für einen bestimmten Index aus.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Integer Wert, der Appertur-Index, der geändert werden soll. (Es gibt die Bereiche 0-4) |
| in | *size* | Pointer auf Double Wert, enthält den Wert des Aperturdurchmessers nach Return. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.13 GOMDMSC15_getAppertureSizeBorders()

```
int __stdcall GOMDMSC15_getAppertureSizeBorders (
             int handle,
             double * minSize,
             double * maxSize )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Liest die die Eingabegrenzen aus, die für setApperturesize verwendet werden können. Diese sind von Gigahertz-Optik festgelegt worden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *minSize* | Pointer auf Double Wert, enthält minimaler Wert des Aperturdurchmessers nach Return. |
| out | *maxSize* | Pointer auf Double Wert, enthält maximaler Wert des Aperturdurchmessers nach Return. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.14 GOMDMSC15_setAppertureIndex()

```
int __stdcall GOMDMSC15_setAppertureIndex (
              int handle,
              int index )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Legt fest, welcher Aperturdurchmesser (index) im Gerät für die Verrechnung verwendet werden soll. Achtung: Der Aperturdurchmesser wird bei den SDK Funktionen nicht automatisch verrechnet. Wenn Sie die Bestrahlungsstärke bestimmen möchten, müssen Sie die Berechnung selbst durchführen.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Integer Wert, der Appertur-Index, der geändert werden soll. (Es gibt die Bereiche 0-4) |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.15 GOMDMSC15_getAppertureIndex()

```
int __stdcall GOMDMSC15_getAppertureIndex (
              int handle,
              int * index )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Legt die Apperturgröße für das MSC15-W fest. Es können bis zu 5 Apperturgrößen im Gerät hinterlegt werden. Mit GOMDMSC15_setAppertureIndex() kann eine Apperur ausgewählt werden.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *index* | Pointer auf Integer Wert, enthält den Appertur-Index nach Return. (Es gibt die Bereiche 0-4) |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.16 GOMDMSC15_setUserCorrectionFactor()

```
int __stdcall GOMDMSC15_setUserCorrectionFactor (
              int handle,
              double value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Legt den vom Benutzer einstellbaren Korrekturfaktor fest. Achtung: Der Korrekturfaktor wird automatisch auf das Spektrum und alle abgeleiteten Messwerte verrechnet. Für die Richtigkeit der Werte übernimmt Gigahertz-Optik keine Garantie.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *value* | Double Wert, der gewünschte Korrekturfaktor. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.17 GOMDMSC15_getUserCorrectionFactor()

```
int __stdcall GOMDMSC15_getUserCorrectionFactor (
            int handle,
            double * value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Liest den eingestellten Korrekturfaktor aus. Achtung: Der Korrekturfaktor wird automatisch auf das Spektrum und alle abgeleiteten Messwerte verrechnet. Für die Richtigkeit der Werte übernimmt Gigahertz-Optik keine Garantie.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Double Wert, enthält den eingestellten Korrekturfaktor nach Return. |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.18 GOMDMSC15_setUserCorrectionFactorLocked()

```
int __stdcall GOMDMSC15_setUserCorrectionFactorLocked (
            int handle,
            bool value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Sperrt den aktuell eingestellten Korrekturfaktor im Gerät. Dieser kann dann im Gerätedisplay vom Benutzer nicht mehr verändert werden.
**Achtung:** Der Korrekturfaktor wird automatisch auf das Spektrum und alle abgeleiteten Messwerte verrechnet. Für die Richtigkeit der Werte übernimmt Gigahertz-Optik keine Garantie.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| in | *value* | Boolean Wert:<br>• true: Wert ist gesperrt<br>• false: Wert kann verändert werden |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.

#### 5.9.2.19 GOMDMSC15_getUserCorrectionFactorLocked()

```
int __stdcall GOMDMSC15_getUserCorrectionFactorLocked (
            int handle,
            bool * value )
```

**Zu beachten**

Diese Funktion ist für die folgenden Gerätetypen gültig. **MSC15-W**
Der Gerätetyp kann mit der Funktion GOMDMSC15_getMSC15DeviceType() ermittelt werden.

Bestimmt, ob der aktuell im Gerät eingestellte Korrekturfaktor gesperrt ist. Achtung: Der Korrekturfaktor wird automatisch auf das Spektrum und alle abgeleiteten Messwerte verrechnet. Für die Richtigkeit der Werte übernimmt Gigahertz-Optik keine Garantie.

**Parameter**

| | | |
|---|---|---|
| in | *handle* | Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten MSC15 & CSS Gerätevariante; dieser Wert wird von der Methode getHandle zurückgeliefert. |
| out | *value* | Pointer auf Boolean Wert:<br>• true: Wert ist gesperrt<br>• false: Wert kann verändert werden |

**Rückgabe**

Rückgabewert: Integer Wert; bei Werten ungleich 0 siehe Rückgabewerttabelle.