Programming manual

S-SDK-BTS2048

Inhaltsverzeichnis

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 BTS2048 der Firma Gigahertz-Optik GmbH mit Hilfe von C/C++ direkt ansteuern zu können. Dies betrifft in erster Linie Kommunikationsbibliotheken und Steuerungsbibliotheken für Ihr BTS2048.
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 D-82299 Türkenfeld Germany Tel.: +49 8193 93700-0 Fax: +49 8193 93700-50 Email: `info@gigahertz-optik.de` Homepage: `http://www.gigahertz-optik.de`	5 Perry Way Newburyport MA 01950 USA Tel: + 978 462 1818 Fax: + 978 462 3677 Email: `b.angelo@gigahertz-optik.com` Homepage: `https://www.gigahertz-optik.de/en-us/home`

1.6 Anforderungen

Für die Benutzung des BTS2048-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 BTS2048-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” 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 BTS2048 mit Ihrem Computers. Die benötigten Treiber sind Standardtreiber von Windows und werden automatisch installiert.

Kapitel 2

Versionsübersicht

Eine Versionsübersicht des S-SDK-BTS2048 folgt:

- **V2014.1**
  Initial version **V2014.2**
  Bugfix: Bei der Initialisierung unter USB konnte es evtl. zu Kommunikationsproblemen kommen
- **V2014.3**
  Bugfix: maximale Anzahl Scans von 10 auf 100 gesetzt
- **V2014.4**
  Bugfix: minimale Schrittweite von 1nm auf 0.25nm gesetzt.
  Bugfix: Speicherlecks behoben
- **V2014.5**
  Bugfix: NoOfScans konnten im Setup nicht gesetzt werden
- **V2014.6**
  Neu: Fehlercode -15003 in Warnung 15003 geändert
  Neu: Fehlercodes / Warnungen in Dokumentation ergänzt bzw. geändert
  Neu: „setColorCalculation“ in Dokumentation aufgenommen
  Neu: „isColorCalculation“ in Dokumentation aufgenommen
  Neu: „saveConfigAsDefault“ in Dokumentation
  Neu: „spectralGetLambdas“ in Dokumentation aufgenommen
- **V2014.7**
  Neu: neuer Fehlercode -15027
  Neu: „getMeasurementMode“ in Dokumentation aufgenommen
  Neu: „getDeltaUV“ in Dokumentation aufgenommen
  Neu: „setDistance“ in Dokumentation aufgenommen
  Neu: „getDistance“ in Dokumentation aufgenommen
  Neu: „getTemperature“ in Dokumentation aufgenommen
  Neu: „spectralSetOffsetMode“ in Dokumentation aufgenommen
  Neu: „spectralGetOffsetMode“ in Dokumentation aufgenommen
  Neu: „spectralMeasureOffset“ in Dokumentation aufgenommen
  Neu: „spectralDeleteOffset“ in Dokumentation aufgenommen
  Bugfix: Speicherleck behoben
- **V2014.8**
  Neu: Performance-Verbesserungen
  Neu: Methode „integralSetIntegrationTimeInMs“
  Neu: Methode „integralGetIntegrationTimeInMs“
  Neu: Methode „getDLLVersion“

Neu: Methode „getFirmwareVersion“
Neu: Methode „getSerialNumber“
Neu: Methode „getType“
Neu: Methode „spectralGetIntegrationTimeRangeInus“
Neu: Methode „integralGetRangeIntegrationTimeRangeInMs“
Neu: Methode „getMaxADC“
Neu: Methode „getNoOfPixels“
Bugfix: Speicherleck behoben
Bugfix: Wenn die Synchronisation zwischen integraler und spektraler Messeinheit aktiviert war, wurde immer eine integrale Messung ausgelöst, auch wenn die integrale Messeinheit deaktiviert war.

- **V2014.9**
  Neu: Performance-Verbesserungen
  Neu: neue Methode „measureGetCountsPixelFast“
  Neu: neue Methode „getDeviceList“
  Neu: neue Fehlercodes / Warncodes

- **V2014.10**
  Neu: Das SDK ist nun passwortgeschützt. Vor Holen eine Handles muss das Freigabepasswort gesetzt werden.
  Neu: Methode setPassword
  Neu: Methode integralGetCurrent
  Neu: Methode spectralEvaluateIntegrationTimeInus
  Neu: Methoden spectralSetDynamicTimeMode / spectralGetDynamicTimeMode
  Neu: Methode getRadiometricValueOverWLRange
  Neu: Methoden zur Kalibrierung (siehe Kapitel: Kalibriermethoden)
  Neu: neue Fehlercodes
  Neu: Methoden setOut1LowDuringMeasurement / getOut1LowDuringMeasurement

- **V2014.11**
  Neu: neue Fehlercodes
  Neu: Performanceverbesserungen

- **V2014.12**
  Neu: Performanceverbesserungen
  Bugfix: in V2014.11 lieferte der integrale Sensor bei nicht synchroner Messung immer „0“.

- **V2014.13**
  Neu: neuer Fehlercode bei Problemen mit der Korrektur mit VL
  Bugfix: in V2014.12 Farbwerte immer „-1“

- **V2014.14**
  Neu: Performanceverbesserungen
  Neu: neue Fehlercodes

- **V2014.15**
  Neu: Methode getLastMaxADC
  Bugfix: Fehlercode alt: -55 -> Fehlercode neu: -15007
  Bugfix: Fehlercode alt: -56 -> Fehlercode neu: -15008

- **V2014.16**
  Neu: Methode spectralSetSavitzkyGolayFilter
  Neu: Methode spectralIsSavitzkyGolayFilter
  Neu: Methode setIPAddress
  Neu: Methode getIPAddress
  Neu: Methode isDHCP
  Neu: Fehlercode -15049

- **V2015.1**
  Neu: Methode isBTS2048VL
  Neu: Methode isBTS2048BS
  Neu: Methode isBTS2048UV

Neu: Methode isBTS2048VLTEC
Neu: Methode setTriggerSource
Neu: Methode getTriggerSource
Neu: Methode setTriggerInternalLevels
Neu: Methode setCooling
Neu: Methode getCoolingState
Neu: Fehlercodes -15051, -15053, -15054, -15055
Neu: Warnungen 15052, 15056

- **V2015.2**
  Neu: Methode integralGetSaturation
  Neu: geänderte Berechnungsmethoden für „scale array with diode“
  Neu: Warnungen 15057, 15058
  Bugfix: underloads und overloads des integralen Sensors wurden manchmal nicht gemeldet

- **V2015.3**
  Neu: Unterstützung neuer Gerätetypen
  Neu: Methode isBTS2048Type

- **V2015.4**
  Neu: weitere Substitutionsmethoden hinzugefügt
  Neu: weitere Kalibriermethoden hinzugefügt
  Neu: Methode setCooling
  Neu: Methode hasCooling
  Neu: Methode isBTS2048Type
  Neu: Methode isStraylightMeasurement
  Neu: Methode isMultiMeasurement
  Neu: Methode spectralSetMeasurementTimeInUs
  Neu: Methode spectralGetMeasurementTimeInUs
  Neu: Methode getMinValidWavelength
  Neu: Methode getMaxValidWavelength
  Neu: Fehlercodes -15029, -15061, -15062, -15063, -15064, -15099, -15100, -15102
  Neu: Warnungen 15025, 15059, 15098, 15103, 15104, 15106

- **V2015.5**
  Bugfix: Performanceverschlechterungen behoben

- **V2015.6**
  Neu: Methoden für substitution Geometrie
  Neu: Methode calibSetWavelengthMapping
  Neu: Methode calibGetWavelengthMapping
  Neu: Methode calibAzSetTransmissionFileActual
  Neu: Fehlercode -15065, -15097

- **V2015.7**
  Bugfix: Anpassung der Saturation bei substitutionMeasurementWithTestDevice und substitution↩ MeasurementWithoutTestDevice
  Bugfix: Methode getMaxADC
  Bugfix: Methode getLastMaxADC

- **V2015.8**
  Neu: Performance-Verbesserungen

- **V2015.9**
  Neu: Methode calibSetCalibMode
  Neu: Methode calibGetCalibMode
  Neu: Methode integralGetLastUsedRange
  Neu: Methode spectralGetSaturation
  Neu: Methode getFilterName
  Neu: Methode setTriggerDelay
  Neu: Methode getTriggerDelay Geändert: Methode getMaxADC Geändert: Methode getLastMaxADC

- **V2016.1**
  Neu: Performance-Verbesserungen
  Neu: Methode spectralMeasurePremeasuredOffset
- **V2016.2**
  Neu: Methode getLastScaleWithVLFactor
  Neu: Methode spectralMeasureOffsetInDarkPosition
- **V2016.3**
  Neu: Methode spectralSaveStaticOffset
  Neu: Methode spectralLoadStaticOffset
- **V2016.6**
  Neu: Methode spectralSetObserver10Degree
  Neu: Methode spectralIsObserver10Degree
- **V2016.9** Bugfix: integrale Kalibrierung
- **V2016.10**
  Neu: PreciseCountCalculation
  Neu: spectralObserver10Degree
- **V2016.11**
  Neu: spectralSetAdvancedNoiseReduction
- **V2016.12**
  Neu: Streulichtmatrix implementiert
- **V2016.13**
  Geändet: Spektrale Kalibrierung und Selbstabsorbtion mit dynamischer Zahl von Mittelungen
  Neu:zusätzliche Warnungen
- **V2016.14**
  Neu: spectralGetLastUsedOffset
- **V2016.15**
  Neu: Calib HighResolutionMode
  Neu: Substitution HighResolutionMode
- **V2017.1**
  Bugfix: external tigger high
- **V2017.2**
  Neu: Routine zur Wellenlängenkalibrierung
- **V2017.3**
  Bugfix: integralGetSaturation
- **V2017.4**
  Neu: Kalibriermodus mit 2 Lampen (z.B. Halogen und Deuterium)
  Neu: integralSetIntegrationTimeInUS und integralGetIntegrationTimeInUS
  Bugfix: HighResolutionMode in der Standardkalibrierung
- **V2017.5**
  Neu: Verbesserung Filterrad Handling
  Bugfix: Messzeiten und Fehlerausgabe bei Standard BP-Messung
- **V2017.6**
  Neu: BP-Messung hochauflösend für UV-Geräte
  Bugfix: Radiometrischer Wert bei Linienlampen
- **V2017.7**
  Neu: Export Premeasured Offset
  Update: Advanced Noise Reduction für UV Geräte
  Bugfix: Multi Messungen

- **V2017.8**
  Neu: TM-30-15
  Update: Angepasste Solar-BP Messung
  Update: TriggerLowDuringMeasurement
- **V2017.9**
  Neu: CIE 170-2
  Bugfix: Selbstabsorbtions Korrektur
- **V2017.10**
  Bugfix: loadConfigFromDevice
- **V2017.11**
  Bugfix: WL-Bereich für die spektrale Kalibration
- **V2018.01**
  Bugfix: setOut1LowDuringMeasurement() FW-Befehl angepasst
- **V2018.02** - V2018.05
  Neu: setDHCPServer und getDHCPServer
  Neu: Initialisierung mit fester IP Adresse in getHandle
  Bugfix: Counts auf Null nach Initialisierung
  Bugfix: Routine im Debug-Logger
  Bugfix: Interpretation des integral Status
- **V2018.06**
  New: Asynchrone Messmethoden
- **V2018.07** - V2018.10
  Update: Performance bei preciseCountCalculation
  Update: Initialisierung mehrerer Geräte über LAN
  Update: Zeit und Temperatur Informationen beim vorgemessenen Offset
  Bugfix: Messstatus für Multi-Messung bei hochauflösendem Messmodus
- **V2018.11**
  Bugfix: Kühlung bei LAN Geräten
- **V2018.12**
  Bugfix: Fehler bei der ersten Vormessung
- **V2018.13**
  Update: zusätzliche Debugging Infos
  Update: Kommunikation / TimeOut LAN
- **V2018.14** - V2018.16
  Update: FWHM, Schwerpunkts und Zentrums-Wellenlänge für BTS2048-UV
  Update: OORSLC Modus Vorgemessen
  Update: Synchronisierungsroutine nacht Kommunikations-Timeout
  Bugfix: Filterpositionszuweisung während der Kalibration
- **V2018.17** - V2018.19
  Neu: Integrale Serienmessung und Ausgang 2 low
  Neu: HTML Dokumentation
  Update: Kalibrierung inklusive Linearitätskorrektur
  Bugfix: Erkennung von spektralem Overload bei NrOfScans $>$ 1
- **V2018.20**
  Neu: Benutzerdefinierte Gewichtungsfunktionen
  Update: Spektraler Offset in Dunkelmodus 0 entfernt

Kapitel 3

Fehlercodes & Warnungen

Eine Auflistung der Fehlercodes und Warnungen folgt.

3.1 Errors

- -15000: Kommunikationsproblem
- -15001: Setup Datei ungültig für BTS2048
- -15002: Setup Datei konnte nicht geöffnet werden
- -15005: Kommunikationskanal kann nicht initialisiert werden
- -15006: zu niedrige Firmwareversion
- -15007: Problem beim Daten Senden
- -15008: Problem beim Daten Empfangen
- -15009: BTS2048 sendet einen nicht näher definierten Fehler
- -15010: delta uv limit < 0
- -15014: error main data eeprom
- -15015: error color data eeprom
- -15016: dieser Befehl ist nicht gültig bei Kommunikation über USB
- -15017: error zero adjust integral amplifier
- -15020: error dark current measurement
- -15026: eine „exception“ wurde abgefangen
- -15029: Eine Einstellung wurde durchgeführt, die bei deaktiviertem integralem Sensor nicht möglich ist
- -15030: Messwert nicht verfügbar, da bei der letzten Messung wurde nicht integral gemessen wurde
- -15031: es sind keine Werte verfügbar
- -15032: es wurde ein falsches Passwort übergeben
- -15033: Kalibrierung: Bewertungsfunktion „Ist“ wurde nicht gesetzt
- -15034: Kalibrierung: Kalibrierlampenspektrum wurde nicht gesetzt

- -15035: Kalibrierung: Kalibriername wurde nicht gesetzt
- -15036: Kalibrierung: spektrale Kalibrierfaktoren wurden nicht gesetzt
- -15037: Kalibrierung: integraler Kalibrierfaktor wurde nicht gesetzt
- -15038: Kalibrierung: spektrale SI-Einheit wurde nicht gesetzt
- -15039: Kalibrierung: integrale SI-Einheit wurde nicht gesetzt
- -15040: Kalibrierung: Filterzuordnung wurde nicht gesetzt
- -15041: Der Wellenlängenbereich wurde zu groß gewählt bzw. die Schrittweiten zu klein, so dass sich eine größere Datenmenge als 3300 Werte ergeben würde
- -15042: Fehler bei der performancetechnischen Vorberechnung
- -15043: Bei der Berechnung der CRI-Werte ist ein Fehler aufgetreten
- -15044: Fehler bei der Berechnung des radiometrischen Wertes über die Wellenlänge
- -15045: Bei der Korrektur mit VL ist ein Fehler aufgetreten.
  Mögliche Ursachen:
  Y gleich „0“ -> wahrscheinlich wurde die Farbberechnung nicht durchgeführt integraler Messwert gleich „0“ -> Fehler bei integraler Messung oder es wurde nicht integral gemessen
- -15047: Zeitüberschreitung bei einer getriggerten Messung
- -15048: Wellenlänge 1 wurde größer oder gleich der Wellenlänge 2 gewählt
- -15049: falsches Format in der IP-Adresse
- -15051: Konfiugrationskonflikt (z.B. statischer Dunkelwert in Kombination mit dynamischer Integrationszeitermittlung)
- -15053: Farbberechnung nicht möglich (z.B. definierte Wellenlängen gehen nicht über den kompletten sichtbaren Bereich)
- -15054: Die augerufene Methode ist für den angeschlossenen Gerätetyp nicht anwendbar
- -15055: Es ist keine externe Stromversorgung angeschlossen
- -15061: Einstellung nicht möglich bei aktiviertem Trigger
- -15062: Einstellung bei deaktivierter spektralen Einheit nicht möglich
- -15063: Einstellung bei statischem dark offset nicht möglich
- -15064: Fehler bei der CQS-Berechnung
- -15065: Einstellung nicht möglich bei dynamischer Messzeitermittlung
- -15067: keine OOR Streulicht Kalibrierung
- -15068: Filterradfehler
- -15069: ungültiges Dunkelsignal
- -15070: OOR Streulicht Korrektur nicht gültig
- -15071: TM-30-15 Berechnungsfehler
- -15072: Firmware Fehler
- -15073: Fehler Flashpage
- -15074: TEC Temperatur Fehler
- -15076: Offset für diese Integrationzeit nicht hinterlegt

- -15077: Fehler beim Lesen der Datei
- -15078: Diese Einstellung ist mit einer virtuellen Kalibrierung nicht möglich
- -15079: Diese Einstellung ist bei einer Kombi Messung nicht möglich
- -15080: Diese Einstellung ist mit einer Streulicht-Kalibrierung nicht möglich
- -15081: Diese Einstellung ist bei hochaufgelöster Messung nicht möglich
- -15082: Diese Einstellung ist nicht mit statischer Integrationszeit möglich
- -15087: Nicht genug Speicher
- -15088: Spektral wurde nicht gemessen
- -15091: Nicht möglich mit dynamischem Dunkelmodus
- -15097: alte aktive Geometrie Substitutionsfaktoren ungültig
- -15098: Kalibrierung nicht gefunden
- -15099: Konfiguration nicht gefunden
- -15100: Parameter außerhalb des zulässigen Bereichs
- -15101: alte aktive DUT Substitutionsfaktoren ungültig
- -15102: Einstellung nicht möglich bei einem BTS2048-UV
- -15997: kein BTS2048 verbunden
- -15998: BTS2048 mit anderer Seriennummer als der erwarteten verbunden
- -15108: Fehler beim Beschreiben des Speichers(GO)
- -15109: Fehler beim beschreiben der Datei.
- -15110: Fehler SLMC Checksumme
- -15111: SLMC Matrix nicht verfügbar
- -15112: SLMC Matrix nicht geladen
- -15115: Asynchrone Messung ist aktiv
- -15116: Fehler asynchrone Messung
- -15117: Messung wurde abgebrochen
- -15118: Nicht alle Lampen wurden gemessen
- -15119: Falsche Kalibrierlampe
- -15120: Keine frei OORSLC Kalibrierung
- -15121: Low Counts Linearisierung nicht verfügbar
- -15122: Fehler LCLinearisierung Checksumme
- -15123: Falscher LCLinearisierungstyp
- -15999: unbekannter Fehler
- -15000: Kommunikationsproblem
- -15001: Setup Datei ungültig für BTS2048
- -15002: Setup Datei konnte nicht geöffnet werden
- -15004: az Modus außerhalb des zulässigen Bereichs (zulässig: 0 - 2)

- -15005: Kommunikationskanal kann nicht initialisiert werden
- -15006: zu niedrige Firmwareversion
- -15007: Problem beim Daten Senden
- -15008: Problem beim Daten Empfangen
- -15009: BTS2048 sendet einen nicht näher definierten Fehler
- -15010: delta uv limit $<$ 0
- -15014: error main data eeprom
- -15015: error color data eeprom
- -15016: dieser Befehl ist nicht gültig bei Kommunikation über USB
- -15017: error zero adjust integral amplifier
- -15020: error dark current measurement
- -15026: eine „exception“ wurde abgefangen
- -15027: Filter nicht gültig für aktuell ausgewählten Kalibriertabelleneintrag
- -15029: Eine Einstellung wurde durchgeführt, die bei deaktiviertem integralem Sensor nicht möglich ist
- -15030: Messwert nicht verfügbar, da bei der letzten Messung wurde nicht integral gemessen wurde
- -15031: es sind keine Werte verfügbar
- -15032: es wurde ein falsches Passwort übergeben
- -15033: Kalibrierung: Bewertungsfunktion „Ist“ wurde nicht gesetzt
- -15034: Kalibrierung: Kalibrierlampenspektrum wurde nicht gesetzt
- -15035: Kalibrierung: Kalibriername wurde nicht gesetzt
- -15036: Kalibrierung: spektrale Kalibrierfaktoren wurden nicht gesetzt
- -15037: Kalibrierung: integraler Kalibrierfaktor wurde nicht gesetzt
- -15038: Kalibrierung: spektrale SI-Einheit wurde nicht gesetzt
- -15039: Kalibrierung: integrale SI-Einheit wurde nicht gesetzt
- -15040: Kalibrierung: Filterzuordnung wurde nicht gesetzt
- -15041: Der Wellenlängenbereich wurde zu groß gewählt bzw. die Schrittweiten zu klein, so dass sich eine größere Datenmenge als 3300 Werte ergeben würde
- -15042: Fehler bei der performancetechnischen Vorberechnung
- -15043: Bei der Berechnung der CRI-Werte ist ein Fehler aufgetreten
- -15044: Fehler bei der Berechnung des radiometrischen Wertes über die Wellenlänge
- -15045: Bei der Korrektur mit VL ist ein Fehler aufgetreten.
  Mögliche Ursachen:
  Y gleich „0“ -$>$ wahrscheinlich wurde die Farbberechnung nicht durchgeführt integraler Messwert gleich „0“ -$>$ Fehler bei integraler Messung oder es wurde nicht integral gemessen
- -15047: Zeitüberschreitung bei einer getriggerten Messung
- -15048: Wellenlänge 1 wurde größer oder gleich der Wellenlänge 2 gewählt
- -15049: falsches Format in der IP-Adresse

- -15051: Konfiugrationskonflikt (z.B. statischer Dunkelwert in Kombination mit dynamischer Integrationszeitermittlung)
- -15053: Farbberechnung nicht möglich (z.B. definierte Wellenlängen gehen nicht über den kompletten sichtbaren Bereich)
- -15054: Die augerufene Methode ist für den angeschlossenen Gerätetyp nicht anwendbar
- -15055: Es ist keine externe Stromversorgung angeschlossen
- -15061: Einstellung nicht möglich bei aktiviertem Trigger
- -15062: Einstellung bei deaktivierter spektralen Einheit nicht möglich
- -15063: Einstellung bei statischem dark offset nicht möglich
- -15064: Fehler bei der CQS-Berechnung
- -15065: Einstellung nicht möglich bei dynamischer Messzeitermittlung
- -15067: keine OOR Streulicht Kalibrierung
- -15068: Filterradfehler
- -15069: ungültiges Dunkelsignal
- -15070: OOR Streulicht Korrektur nicht gültig
- -15071: TM-30$*$ - -15 Berechnungsfehler
- -15072: Firmware Fehler
- -15073: Fehler Flashpage
- -15074: TEC Temperatur Fehler
- -15076: Offset für diese Integrationzeit nicht hinterlegt
- -15077: Fehler beim Lesen der Datei
- -15078: Diese Einstellung ist mit einer virtuellen Kalibrierung nicht möglich
- -15079: Diese Einstellung ist bei einer Kombi Messung nicht möglich
- -15080: Diese Einstellung ist mit einer Streulicht-Kalibrierung nicht möglich
- -15081: Diese Einstellung ist bei hochaufgelöster Messung nicht möglich
- -15082: Diese Einstellung ist nicht mit statischer Integrationszeit möglich
- -15087: Nicht genug Speicher
- -15088: Spektral wurde nicht gemessen
- -15091: Nicht möglich mit dynamischem Dunkelmodus
- -15092: nur bei USB Verbindung möglich
- -15093: Fehler Dunkelstrom Messung
- -15097: alte aktive Geometrie Substitutionsfaktoren ungültig
- -15098: Kalibrierung nicht gefunden
- -15099: Konfiguration nicht gefunden
- -15100: Parameter außerhalb des zulässigen Bereichs
- -15101: alte aktive DUT Substitutionsfaktoren ungültig
- -15102: Einstellung nicht möglich bei einem BTS2048-UV

- -15997: kein BTS2048 verbunden
- -15998: BTS2048 mit anderer Seriennummer als der erwarteten verbunden
- -15108: Fehler beim Beschreiben des Speichers(GO)
- -15109: Fehler beim beschreiben der Datei.
- -15110: Fehler SLMC Checksumme
- -15111: SLMC Matrix nicht verfügbar
- -15112: SLMC Matrix nicht geladen
- -15115: Asynchrone Messung ist aktiv
- -15116: Fehler asynchrone Messung
- -15117: Messung wurde abgebrochen
- -15118: Nicht alle Lampen wurden gemessen
- -15119: Falsche Kalibrierlampe
- -15120: Keine frei OORSLC Kalibrierung
- -15121: Low Counts Linearisierung nicht verfügbar
- -15122: Fehler LCLinearisierung Checksumme
- -15123: Falscher LCLinearisierungstyp
- -15999: unbekannter Fehler

3.2 Warnings

- 15003: file not found: es wurden bislang keine Defaultdaten gespeichert. Daher existiert auch keine Defaultdatei
- 15011: die integrale Einheit meldet einen „overload“
- 15012: die integrale Einheit meldet einen underload
- 15023: die spektrale Einheit meldet einen “overload”
- 15025: die spektrale Einheit meldet einen „underload“
- 15028: die Integrationszeit für die integrale Einheit wurde an das zulässige Raster angepasst
- 15046: wenn der dark mode auf „static“ steht und die Integrationszeit der spektralen Einheit auf „dynamische Ermittlung“ umgeschaltet wird, wird der „dark mode“ automatisch auf „dynamic“ geändert, da in diesem Fall der dark mode „static“ nicht zulässig ist
- 15052: Die Farbberechnung wurde abgeschaltet, da der definierte Wellenlängenbereich nicht den kompletten sichtbaren Bereich (380nm – 780nm) abdeckt
- 15056: Die Integrationszeit wurde angepasst, da die Kühlung ausgeschaltet wurde
- 15057: die spektrale Einheit meldet „overload“, die integrale Einheit meldet „overload“
- 15058: die spektrale Einheit meldet „overload“, die integrale Einheit meldet „underload“
- 15059: Das Subsitutionskorrekturmodul hat festgestellt, dass zwischen vorheriger und aktueller Messung eine andere Integrationszeit verwendet wurde. Beide notwendigen Messungen müssen mit der gleichen Integrationszeit durchgeführt werden.

- 15060: Undefinierter Integraler Fehler
- 15066: OOR Modus wurde geändert
- 15075: Falsche Filterposition für Messung
- 15083: Zeitermittlung der Integrationszeit wurde geändert.
- 15084:Messmodus mit hoher Auflösung wurde geändert.
- 15085: Messung dauert über 10 Minuten
- 15086: Anzahl Mittelungen sehr niedrig ($<$10)
- 15089: undefinierter spektraler Fehler
- 15090: Integraler Bereich wurde geändert
- 15094: Array wenig Signal und integraler Overload
- 15095: Array wenig Signal und integraler Underload
- 15098: Beim Durchsuchen des Konfigurationsspeichers wurde auf einen Index zugegriffen, der nicht belegt ist.
- 15103: Der eingestellte a(z)-mode (spectral mismatch correction) ist mit der letzten aufgerufenen Einstellung nicht kompatibel und wurde automatisch angepasst.
- 15104: Die Anpassung der spektralen Daten aufgrund einer integralen Messung wurde ausgeschaltet, da der integrale Sensor deaktiviert wurde.
- 15105: Trigger deaktiviert
- 15106: Triggerquelle wurde automatisch von intern (Triggerung durch integralen Level) auf extern geändert, da der integrale Sensor deaktiviert wurde.
- 15107: Kühlung nicht in Ordnung
- 15113: SLMC Matrix muss geldaen werden
- 15114: keine asynchrone Messung möglich

Kapitel 4

Beispiel - SDK in Applikation einbinden

Zu beachten

Beispiele wie die DLL´s in die Applikation eingebunden werden können befinden sich auf der mitgelieferten CD und in der html-Dokumentation.
Unter den einzelenen Kapitel der Methodenbeschreibungen (Moduls) befinden sich seperate Beispiele wie die Methode zu verwenden sind.

4.1 C++

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 BTS2048, welches per USB oder LAN verbunden ist, führt eine Messung durch und schreibt die Ergebnisse auf die Konsole.
Am Ende werden alle BTS2048 - Resourcen wieder freigegeben.

4.1.1 BTS2048Example.cpp

```
#include "BTS2048Import.h"
#include <iostream>

int main(int argc, char* argv[])
{
    BTS2048Import bts2048;

    //search for a BTS2048 device
    //first you have to replace the right password in the BTS2048Import.cpp
    int error = bts2048.init("BTS2048_0");
    if (error == 0)
    {
        char userinput[10];
        //write all available calibration entries to the console
        bts2048.writeCalibrationInfoToConsole();

        //let the user choose a calibration
        std::cout << "Please choose a calibration number:";
        std::cin.getline(userinput, 10);
        bts2048.setCalibrationEntry(atoi(userinput));

        //set measurement mode and start a new measurement
        //dynamicTimeMode = true
        //offsetMode = 0
        //spectralIntegrationTime = 50ms
        bts2048.setSpectralMeasurementMode(true, 0, 50000);
        error = bts2048.measure();

        //if no error occured read the integral values
        if (error == 0)
```

```
        {
            double value;
            char unit[2048];
            bts2048.integralGetValues(&value, unit);
            std::cout << "integral sensor = " << value << " " << unit << std::endl;
        }
        else
        {
            std::cout << "error occured: " << error << std::endl;
        }
        bts2048.close();
    }
    else
    {
        std::cout << "error occured: " << error << std::endl;
    }
    system("PAUSE");
}
```

4.1.2 BTS2048Import.cpp

```
#include "BTS2048Import.h"

BTS2048Import::BTS2048Import()
{
    hDLLGOBTS2048 = NULL;
    handle = -1;
}

BTS2048Import::~BTS2048Import()
{
}

int __stdcall BTS2048Import::init(char* deviceName)
{
    int l_rc = 0;
    if (handle > 0)
        close();
    if (getProcAddresses(&hDLLGOBTS2048, "GOMDBTS2048.dll", 12,
        &GOMDBTS2048_setPassword, "GOMDBTS2048_setPassword",
        &GOMDBTS2048_getHandle, "GOMDBTS2048_getHandle",
        &GOMDBTS2048_releaseHandle, "GOMDBTS2048_releaseHandle",
        &GOMDBTS2048_setCalibrationEntryNumber, "
      GOMDBTS2048_setCalibrationEntryNumber",
        &GOMDBTS2048_getSelectedCalibrationEntryNumber, "
      GOMDBTS2048_getSelectedCalibrationEntryNumber",
        &GOMDBTS2048_readCalibrationEntryInfo, "
      GOMDBTS2048_readCalibrationEntryInfo",
        &GOMDBTS2048_measure, "GOMDBTS2048_measure",
        &GOMDBTS2048_getCWValue, "GOMDBTS2048_getCWValue",
        &GOMDBTS2048_integralGetUnit, "GOMDBTS2048_integralGetUnit",
        &GOMDBTS2048_spectralSetDynamicTimeMode, "
      GOMDBTS2048_spectralSetDynamicTimeMode",
        &GOMDBTS2048_spectralSetOffsetMode, "
      GOMDBTS2048_spectralSetOffsetMode",
        &GOMDBTS2048_spectralSetIntegrationTimeInus, "
      GOMDBTS2048_spectralSetIntegrationTimeInus"
        ))
    {
        try {
            l_rc = GOMDBTS2048_setPassword("passw"); //replace passw with the right
       password
            if (l_rc == 0)
                l_rc = GOMDBTS2048_getHandle(deviceName, &handle);
        }
        catch (...) {
            l_rc = -1;
        }
    }
    else {
        l_rc = -1;
    }
    return l_rc;
}

int __stdcall BTS2048Import::writeCalibrationInfoToConsole()
{
    char calibInfo[100];
    std::cout << "Available calibration entries:" << std::endl;
    for (int i = 0; i < 52; i++)
    {
        GOMDBTS2048_readCalibrationEntryInfo(handle, i, calibInfo);
```

```
        if (*calibInfo != '\0')
        {
            std::cout << i << ": " << calibInfo << std::endl;
        }
    }
    return 0;
}

int __stdcall BTS2048Import::setCalibrationEntry(int value)
{
    int l_rc = GOMDBTS2048_setCalibrationEntryNumber(handle, value);
    return l_rc;
}

int __stdcall BTS2048Import::setSpectralMeasurementMode(bool dynamicTimeMode, int offsetMode, int
  integrationtime)
{
    int l_rc = GOMDBTS2048_spectralSetDynamicTimeMode(handle,
      dynamicTimeMode);
    if (l_rc < 0)
        return l_rc;

    l_rc = GOMDBTS2048_spectralSetOffsetMode(handle, offsetMode);
    if (l_rc < 0)
        return l_rc;

    l_rc = GOMDBTS2048_spectralSetIntegrationTimeInus(handle,
      integrationtime);
    return l_rc;
}

int __stdcall BTS2048Import::measure()
{
    int l_rc = GOMDBTS2048_measure(handle);
    return l_rc;
}

int __stdcall BTS2048Import::integralGetValues(double* value, char* unit)
{
    int l_rc = GOMDBTS2048_getCWValue(handle, value);
    if (l_rc < 0)
        return l_rc;

    int calibrationEntryNumber;
    l_rc = GOMDBTS2048_getSelectedCalibrationEntryNumber(
      handle, &calibrationEntryNumber);
    if (l_rc < 0)
        return l_rc;

    l_rc = GOMDBTS2048_integralGetUnit(handle, calibrationEntryNumber, unit);
    return l_rc;
}

int __stdcall BTS2048Import::close()
{
    int l_rc = GOMDBTS2048_releaseHandle(handle);
    handle = -1;
    return l_rc;
}

bool __stdcall BTS2048Import::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;
}
```

4.1.3 BTS2048Import.h

```
#ifndef BTS2048ImportH
#define BTS2048ImportH

#include <Windows.h>
#include "stdio.h"
#include <iostream>

class BTS2048Import
{
public:
    BTS2048Import();
    virtual ~BTS2048Import();
    int __stdcall init(char* deviceName);
    int __stdcall close();
    int __stdcall writeCalibrationInfoToConsole();
    int __stdcall setCalibrationEntry(int value);
    int __stdcall setSpectralMeasurementMode(bool dynamicTimeMode, int offsetMode, int integrationtime);
    int __stdcall integralGetValues(double* value, char* unit);
    int __stdcall measure();

private:
    int handle;
    HINSTANCE hDLLGOBTS2048;
    bool __stdcall getProcAddresses(HINSTANCE *p_hLibrary, const char* p_dllName, int p_count, ...);

    int(__stdcall *GOMDBTS2048_setPassword)(char* password);
    int(__stdcall *GOMDBTS2048_getHandle)(char* deviceName, int* handle);
    int(__stdcall *GOMDBTS2048_releaseHandle)(int handle);

    int(__stdcall *GOMDBTS2048_setCalibrationEntryNumber)(int handle,
      int calibrationEntryNumber);
    int(__stdcall *GOMDBTS2048_getSelectedCalibrationEntryNumber
      )(int handle, int* calibrationEntryNumber);
    int(__stdcall *GOMDBTS2048_readCalibrationEntryInfo)(int handle,
      int calibrationEntryNumber, char* calibrationName);

    int(__stdcall *GOMDBTS2048_spectralSetDynamicTimeMode)(int handle
      , bool value);
    int(__stdcall *GOMDBTS2048_spectralSetOffsetMode)(int handle, int
      value);
    int(__stdcall *GOMDBTS2048_spectralSetIntegrationTimeInus)(
      int handle, int timeInus);

    int(__stdcall *GOMDBTS2048_measure)(int handle);
    int(__stdcall *GOMDBTS2048_getCWValue)(int handle, double* value);
    int(__stdcall *GOMDBTS2048_integralGetUnit)(int handle, int
      calibrationEntryNumber, char* unit);
};
#endif
```

4.2 Weitere Beispiel

Weitere Beispiele zum Einbinden der DLL´s in delphi, pyton, java, etc. befinden sich im Installationsverzeichnis der SDK.

Kapitel 5

Modul-Dokumentation

5.1 Informationen zu den Methoden

Alle hier beschrieben Methoden können bei jedem BTS2048 angewendet warden. 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 in der Dokumentation.

5.2 Standard SDK Methoden

Methoden für die Initalisierung des BTS2048 und allgemeinen Verwendung des SDKs.

Funktionen

- int __stdcall GOMDBTS2048_setPassword (char *value)
- int __stdcall GOMDBTS2048_getDLLVersion (char *value)
- void __stdcall GOMDBTS2048_getDeviceList (int commType, char *values[ ], int listSize)
- int __stdcall GOMDBTS2048_getHandle (char *device, int *handle)
- int __stdcall GOMDBTS2048_releaseHandle (int handle)
- int __stdcall GOMDBTS2048_getFirmwareVersion (int handle, char *value)
- int __stdcall GOMDBTS2048_getSerialNumber (int handle, char *value)
- int __stdcall GOMDBTS2048_getType (int handle, char *value)
- int __stdcall GOMDBTS2048_isBTS2048VL (int handle, bool *value)
- int __stdcall GOMDBTS2048_isBTS2048BS (int handle, bool *value)
- int __stdcall GOMDBTS2048_isBTS2048UV (int handle, bool *value)
- int __stdcall GOMDBTS2048_isBTS2048VLTEC (int handle, bool *value)
- int __stdcall GOMDBTS2048_isBTS2048Type (int handle, int type, bool *value)
- int __stdcall GOMDBTS2048_setCooling (int handle, bool value)
- int __stdcall GOMDBTS2048_getCoolingState (int handle, int *value)
- int __stdcall GOMDBTS2048_hasCooling (int handle, bool *value)
- int __stdcall GOMDBTS2048_getMaxADC (int handle, int *value)
- int __stdcall GOMDBTS2048_getNoOfPixels (int handle, int *value)
- int __stdcall GOMDBTS2048_setIPAddress (int handle, char *value)
- int __stdcall GOMDBTS2048_getIPAddress (int handle, char *value)
- int __stdcall GOMDBTS2048_isDHCP (int handle, bool *value)
- int __stdcall GOMDBTS2048_setDHCPServer (int handle, bool value)
- int __stdcall GOMDBTS2048_isDHCPServer (int handle, bool *value)
- int __stdcall GOMDBTS2048_isConnected (int handle, bool *value)

5.2.1 Ausführliche Beschreibung

5.2.2 C++ Aufrufbeispiel

Das folgende Beispiel zeigt die Initialisierung des BTS2048. Dies muss bei jeder Inbetriebnahme durchgeführt werden.

```
GOMDBTS2048_setPassword("Your password");
int handle;
int l_rc = GOMDBTS2048_getHandle("BTS2048_0", &handle);        //initialization of
       first found @device
if (handle > 0 )
{
        // do something
}
GOMDBTS2048_releaseHandle(handle);                    //release handle
```

5.2.3 Dokumentation der Funktionen

5.2.3.1 GOMDBTS2048_setPassword()

```
int __stdcall GOMDBTS2048_setPassword (
            char * value )
```

Diese Methode muss vor allen anderen aufgerufen werden, um die Benutzung des SDK freizuschalten. Die Freischaltung erfolgt auf mehreren Ebenen:

- Ebene 1: Benutzung des SDK im Allgemeinen
- Ebene 2: Alle Elemente der 1. Ebene plus die Speicherung von Kalibrierungen im benutzerspezifischen Speicherbereich
- Ebene 3: Alle Elemente der 2. Ebene plus die Speicherung von Kalibrierungen im Orginal Bereich (Rekalibrierung) Die Passwörter werden Ihnen 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 GOMDBTS2048_getDLLVersion()

```
int __stdcall GOMDBTS2048_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 GOMDBTS2048_getDeviceList()

```
void __stdcall GOMDBTS2048_getDeviceList (
            int commType,
            char * values[],
            int listSize )
```

Liefert sämtliche im System verfügbaren Geräte

- Per USB angeschlossene BTS2048 liefern folgenden Output: „BTS2048;Serial:<Seriennummer>;USB“
- Per LAN angeschlossene BTS2048 liefern folgenden Output:„BTS0248;Serial:<Seriennummer>;LAN;IP↩ :<IP-Adresse>“ <Seriennummer> entspricht der Seriennummer des BTS2048, <IP-Adresse> entspricht der tatsächlichen IP-Adresse im Netzwerk Zur Verwendung muss ein String-Array vordefiniert werden in welchem die gefundenen Gerätedefinitionen abgelegt werden. Ist das String-Array zu klein werden evtl. nicht alle vorhandenen Geräte angezeigt. Nicht verwendete Arraypositionen werden mit einem Leerstring bestückt. Die Größe jeder einzelnen Listenposition beträgt 50 Zeichen.

Parameter

in	*commType*	Integer-Wert: • -1: Alle Geräte unabhängig von der angeschlossenen Kommunikationsschnittstelle • 0: Nur per USB angeschlossene Geräte • 1: Nur per LAN angeschlossene Geräte
out	*values*	Array von Strings; enthält nach Rücksprung alle gefundenen BTS2048.
out	*listSize*	Integer-Wert, entspricht der Größe der zurückgelieferten Geräteliste.

Rückgabe

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

5.2.3.4 GOMDBTS2048_getHandle()

```
int __stdcall GOMDBTS2048_getHandle (
             char * device,
             int * handle )
```

Nach der Freischaltung des SDK muss grundsätzlich diese Methode als nächstes aufgerufen werden, um das B↩ TS2048 zu initialisieren. Der Parameter „handle“ beinhaltet eine eindeutige Zuordnungsnummer zum instanziierten Messgerät, die beim Aufruf aller weiteren Methoden als erster Parameter übergeben werden muss.
Nachdem der erste Handle gefunden wurde, liefert der nächste Aufruf von getHandle(BTS2048_0, &handle), das nächste angeschlossene BTS2048, sofern der Handle des ersten Geräts nicht wieder freigegeben wurde.

Parameter

in	*device*	Nullterminierter String, der das gewünschte zu initialisierende Gerät kennzeichnet. Der String hat immer folgenden Aufbau: “BTS2048_<serial>”. <serial> steht als Platzhalter für die Seriennummer des Messgeräts. Der String „BTS2048_5678“ initialisiert z.B. das BTS2048 mit der Seriennummer 5678. Eine weitere Möglichkeit ist die Übergabe von NULL. Damit wird das erste unter Windows registrierte BTS2048 initialisiert. Falls Sie ein Gerät über LAN mit einer bestimmten IP Adresse (siehe setIPAdress()) initialisieren wollen, können Sie den Term "_IPXXX.XXX.XXX.XXX" an den Initialisierungs-String hängen (z.B. "BTS2048_0_IP192.168.002.074")
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.

5.2.3.5 GOMDBTS2048_releaseHandle()

```
int __stdcall GOMDBTS2048_releaseHandle (
            int handle )
```

Diese Methode muss zum Abschluss aufgerufen werden, um die vom BTS2048 belegten Resourcen/ Speicher wieder freizugeben.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.2.3.6 GOMDBTS2048_getFirmwareVersion()

```
int __stdcall GOMDBTS2048_getFirmwareVersion (
            int handle,
            char * value )
```

Liefert die Firmware-Version des verbundenen BTS2048.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; 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 GOMDBTS2048_getSerialNumber()

```
int __stdcall GOMDBTS2048_getSerialNumber (
            int handle,
            char * value )
```

Liefert die Seriennummer des verbundenen BTS2048.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Nullterminierter String; enthält nach Rücksprung die Seriennummer des BTS2048, Mindestgröße: 10 Bytes

Rückgabe

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

5.2.3.8 GOMDBTS2048_getType()

```
int __stdcall GOMDBTS2048_getType (
            int handle,
            char * value )
```

Ermittelt den angeschlossenen BTS2048 Gerätetyp.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Nullterminierter String; enthält nach Rücksprung den Typ des BTS2048, Mindestgröße: 30 Bytes

Rückgabe

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

5.2.3.9 GOMDBTS2048_isBTS2048VL()

```
int __stdcall GOMDBTS2048_isBTS2048VL (
            int handle,
            bool * value )
```

Evaluiert, ob das angeschlossene BTS2048 vom Typ BTS2048-VL ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung die Information, ob das angeschlossene Messgerät vom Typ BTS2048-VL ist. • true: Gerät ist vom Typ BTS2048-VL • false: Gerät ist nicht vom Typ BTS2048-VL

Rückgabe

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

5.2.3.10 GOMDBTS2048_isBTS2048BS()

```
int __stdcall GOMDBTS2048_isBTS2048BS (
            int handle,
            bool * value )
```

Evaluiert, ob das angeschlossene BTS2048 vom Typ BTS2048-BS ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung die Information, ob das angeschlossene Messgerät vom Typ BTS2048-BS ist. • true: Gerät ist vom Typ BTS2048-VL • false: Gerät ist nicht vom Typ BTS2048-BS

Rückgabe

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

5.2.3.11 GOMDBTS2048_isBTS2048UV()

```
int __stdcall GOMDBTS2048_isBTS2048UV (
            int handle,
            bool * value )
```

Evaluiert, ob das angeschlossene BTS2048 vom Typ BTS2048-UV ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung die Information, ob das angeschlossene Messgerät vom Typ BTS2048-UV ist. • true: Gerät ist vom Typ BTS2048-VLTec • false: Gerät ist nicht vom Typ BTS2048-VLTec

Rückgabe

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

5.2.3.12 GOMDBTS2048_isBTS2048VLTEC()

```
int __stdcall GOMDBTS2048_isBTS2048VLTEC (
            int handle,
            bool * value )
```

Evaluiert, ob das angeschlossene BTS2048 vom Typ BTS2048-VLTec ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung die Information, ob das angeschlossene Messgerät vom Typ BTS2048-VLTec ist. • true: Gerät ist vom Typ BTS2048-VLTec • false: Gerät ist nicht vom Typ BTS2048-VLTec

Rückgabe

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

5.2.3.13 GOMDBTS2048_isBTS2048Type()

```
int __stdcall GOMDBTS2048_isBTS2048Type (
            int handle,
            int type,
            bool * value )
```

Evaluiert, ob das angeschlossene BTS2048 von einem spezifischen Typ ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*type*	Integer Wert, dieser definiert den Gerätetype der abgeprüft werden soll. 0: VL, 1: UV, 2: BS, 3: VL-TEC, 4: UV-S, 5: UV-S-WP, 6: VL-F, 7: VL-F-TEC
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung die Information, ob das angeschlossene Messgerät vom spezifizierten Typ ist • true: Gerät ist vom spezifizierten Typ • false: Gerät ist nicht vom spezifizierten Typ

Rückgabe

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

5.2.3.14 GOMDBTS2048_setCooling()

```
int __stdcall GOMDBTS2048_setCooling (
             int handle,
             bool value )
```

Mit dieser Methode kann die Kühlung des Messgerätes eingeschaltet werden. Dies ist nur beim Gerätetyp BTS2048-VLTec möglich. Im eingeschalteten Zustand kann die spektrale Messzeit bis zu 60 Sekunden betragen. Im ausgeschalteten Zustand sind nur maximal 4 Sekunden Messzeit zulässig.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean-Wert: • true: Kühlung an • false: Kühlung aus

Rückgabe

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

5.2.3.15 GOMDBTS2048_getCoolingState()

```
int __stdcall GOMDBTS2048_getCoolingState (
             int handle,
             int * value )
```

Diese Methode ermittelt, ob die Kühlung ein- bzw. ausgeschaltet ist. Falls die Kühlung aktiv ist, wird zusätzlich ermittelt, ob die Temperatur für eine stabile Messung in Ordnung ist.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer-Wert: • 0: Kühlung aus • 1: Kühlung an, Temperatur nicht in Ordnung • 2: Kühlung an Temperatur in Ordnung

Rückgabe

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

5.2.3.16 GOMDBTS2048_hasCooling()

```
int __stdcall GOMDBTS2048_hasCooling (
            int handle,
            bool * value )
```

Diese Methode ermittelt, ob das angeschlossene BTS2048 Gerät eine interne Kühlung hat.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • false: Keine Kühlung • true: Kühlung vorhanden

Rückgabe

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

5.2.3.17 GOMDBTS2048_getMaxADC()

```
int __stdcall GOMDBTS2048_getMaxADC (
            int handle,
            int * value )
```

Liefert die maximal mögliche Anzahl an Counts der spektralen Messeinheit. Ab Firmwareversion 1.42 wird ein Offset abgezogen um die tatsächlich nutzbaren Counts des ADCs zu wiederzugeben. Dadurch ist der Rückgabewert kleiner als die tatsächlich vorhandenen 16bit.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer, enthält nach Rücksprung die maximal mögliche Anzahl an Counts der spektralen Messeinheit.

Rückgabe

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

5.2.3.18 GOMDBTS2048_getNoOfPixels()

```
int __stdcall GOMDBTS2048_getNoOfPixels (
            int handle,
            int * value )
```

Liefert die Pixelanzahl der spektralen Messeinheit.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer, enthält nach Rücksprung die Pixelanzahl der spektralen Messeinheit.

Rückgabe

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

5.2.3.19 GOMDBTS2048_setIPAddress()

```
int __stdcall GOMDBTS2048_setIPAddress (
            int handle,
            char * value )
```

Diese Methode setzt die IP Adresse im BTS2048 im Format „000.000.000.000“. Führende Nullen können vernachlässigt werden. Im Falle eines Formatfehlers (z.B. Adressteile mit Werten $>$ 255 oder $<$ 0, mehr oder weniger Adressteile, fehlerhafte Zeichen) wird ein Fehlercode zurückgegeben. Wenn die IP Adresse „000.000.000.000“ übergeben wird, wird das Gerät in den DHCP Modus gesetzt. Es versucht also, sich bei der Initialisierung des L↩ AN-Interfaces eine IP Adresse von einem DHCP-Server zu holen. Wenn sich das Gerät im DHCP-Modus befindet, jedoch keine IP-Adresse durch einen externen DHCP-Server erhalten konnte, vergibt der interne DHCP-Server die Adresse 169.254.1.1. Ein per LAN-Kabel direkt verbundener PC bekommt ebenfalls eine Adress im Adressbereich 169.254.1.1 – 169.254.255.255
Die Änderung der IP Adresse wird erst nach Neustart des BTS2048 aktiv. Dazu muss das Gerät vom Stromnetz entfernt werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Null-terminated string; z.B. „192.168.178.25“

Rückgabe

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

5.2.3.20 GOMDBTS2048_getIPAddress()

```
int __stdcall GOMDBTS2048_getIPAddress (
            int handle,
            char * value )
```

Diese Methode liefert die aktuelle IP-Adresse zurück.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Nullterminierter String, muss mit 16 Bytes allokiert werden und enthält nach Rücksprung die aktuelle tatsächliche IP Adresse.

Rückgabe

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

5.2.3.21 GOMDBTS2048_isDHCP()

```
int __stdcall GOMDBTS2048_isDHCP (
            int handle,
            bool * value )
```

Diese Methode gibt an, ob die IP-Adresse des Geräts per DHCP zugewiesen werden soll oder eine fixe IP Adresse vergeben wird. Achtung: Diese Methode sagt nichts darüber aus, ob der DHCP Server des Gerätes an- oder abgeschaltet ist. Benutzen SIe hierfür die Funktion: isDHCPServer()

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung den Status: • false: Feste IP-Adresse im Gerät hinterlegt • true: IP-Adresse wird per DHCP zugewiesen

Rückgabe

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

5.2.3.22 GOMDBTS2048_setDHCPServer()

```
int __stdcall GOMDBTS2048_setDHCPServer (
            int handle,
            bool value )
```

Mit dieser Methode kann der Status des DHCP Servers gesetzt werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Status des DHCP Server: true = aktiv, false = inaktiv

Rückgabe

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

5.2.3.23 GOMDBTS2048_isDHCPServer()

```
int __stdcall GOMDBTS2048_isDHCPServer (
```

```
            int handle,
            bool * value )
```

Diese Methode liefert die Information, ob der DHCP Server das Gerät aktiv ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung den Status des DHCP Servers:

Rückgabe

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

5.2.3.24 GOMDBTS2048_isConnected()

```
int __stdcall GOMDBTS2048_isConnected (
            int handle,
            bool * value )
```

Diese Methode überprüft ob das Gerät noch mit dem PC verbunden ist oder getrennt wurde

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung den Status der Verbindung.

Rückgabe

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

5.3 Methoden für allgemeine Messeinstellungen

Allgemeine Einstellungen um eine Messung durchzuführen.

Funktionen

- int __stdcall GOMDBTS2048_saveConfig (int handle, char ∗filename)
- int __stdcall GOMDBTS2048_loadConfig (int handle, char ∗filename)
- int __stdcall GOMDBTS2048_saveConfigAsDefault (int handle)
- int __stdcall GOMDBTS2048_setCalibrationEntryNumber (int handle, int calibrationEntryNumber)
- int __stdcall GOMDBTS2048_getSelectedCalibrationEntryNumber (int handle, int ∗calibrationEntryNumber)
- int __stdcall GOMDBTS2048_readCalibrationEntryInfo (int handle, int calibrationEntryNumber, char ∗calibrationName)
- int __stdcall GOMDBTS2048_getMeasurementQuantity (int handle, int calibrationEntryNumber, char ∗quantity)
- int __stdcall GOMDBTS2048_isMeasurementQuantity (int handle, int calibrationEntryNumber, char ∗quantity, bool ∗isQuantity)
- int __stdcall GOMDBTS2048_getSelectedMeasurementQuantity (int handle, char ∗quantity)
- int __stdcall GOMDBTS2048_isMultiMeasurement (int handle, int calibrationEntryNumber, bool ∗value)
- int __stdcall GOMDBTS2048_isOORSLCorrectionMeasurement (int handle, int calibrationEntryNumber, bool ∗value)
- int __stdcall GOMDBTS2048_isSLMCorrectionMeasurement (int handle, int calibrationEntryNumber, bool ∗value)
- int __stdcall GOMDBTS2048_setMeasurementMode (int handle, int measurementMode)
- int __stdcall GOMDBTS2048_getMeasurementMode (int handle, int ∗measurementMode)
- int __stdcall GOMDBTS2048_setSpectralIntegralSynch (int handle, bool value)
- int __stdcall GOMDBTS2048_isSpectralIntegralSynch (int handle, bool ∗value)
- int __stdcall GOMDBTS2048_setDistance (int handle, double distance)
- int __stdcall GOMDBTS2048_getDistance (int handle, double ∗distance)
- int __stdcall GOMDBTS2048_getFilterName (int handle, int position, char ∗value)
- int __stdcall GOMDBTS2048_getFilterNameforCalibration (int handle, int calibrationEntryNumber, char ∗value)

5.3.1 Ausführliche Beschreibung

5.3.2 C++ Aufrufbeispiel

Momentane Gerätekonfiguration von BTS2048 abspeichern.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);          //initialization
GOMDBTS2048_saveConfig(handle, "C:\\temp\\configfile.dat");       //saves the
       configuration of your BTS2048
GOMDBTS2048_releaseHandle(handle);             //release handle
```

5.3.3 Dokumentation der Funktionen

5.3.3.1 GOMDBTS2048_saveConfig()

```
int __stdcall GOMDBTS2048_saveConfig (
             int handle,
             char * filename )
```

Die aktuell gesetzten Parameter werden in einer Konfigurationsdatei zur späteren Wiederverwendung gespeichert. Die Werte können mit „loadConfig“ wieder geladen werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*filename*	nullterminierter String; Dateiname inkl. Pfad unter dem die Konfigurationsdaten abgelegt werden sollen.

Rückgabe

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

5.3.3.2 GOMDBTS2048_loadConfig()

```
int __stdcall GOMDBTS2048_loadConfig (
              int handle,
              char * filename )
```

Diese Methode lädt alle zuvor gesetzten und gespeicherten Werte aus der spezifizierten Datei. Wenn die Konfigurationsdatei nicht zu einem BTS2048, sondern zu einem anderen Gerät gehört, wird ein Fehlercode als Rückgabewert geliefert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*filename*	Kompletter Pfad zu einer Konfigurationsdatei, in welcher zuvor bestehende Einstellungen gespeichert wurden.

Rückgabe

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

5.3.3.3 GOMDBTS2048_saveConfigAsDefault()

```
int __stdcall GOMDBTS2048_saveConfigAsDefault (
              int handle )
```

Die aktuell gesetzten Parameter werden in einer Konfigurationsdatei zur späteren Wiederverwendung gespeichert und bei Neuinitialisierung des Gerätes wieder eingeladen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.3.3.4 GOMDBTS2048_setCalibrationEntryNumber()

```
int __stdcall GOMDBTS2048_setCalibrationEntryNumber (
            int handle,
            int calibrationEntryNumber )
```

Das BTS2048 wird mit einer oder mehreren Kalibrierungen ausgeliefert. Diese dienen unterschiedlichen Messszenarien. Mit dieser Methode können Sie die im Eeprom abgelegten Kalibrierungen selektieren. Es exisiteren bis zu 52 Kalibriereinträge, von denen nicht jeder Speicherplatz mit einem Eintrag belegt sein muss. Falls ein nicht existierender Kalibrierindex ausgewählt wird, liefert die Methode einen Fehlercode.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, Wertebereich 0-51, nicht belegte Kalibriertabelleneinträge liefern eine Fehlermeldung.

Rückgabe

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

5.3.3.5 GOMDBTS2048_getSelectedCalibrationEntryNumber()

```
int __stdcall GOMDBTS2048_getSelectedCalibrationEntryNumber (
            int handle,
            int * calibrationEntryNumber )
```

Diese Methode liefert den selektierten Kalibriertabellenindex. Dieser kann bei Methoden wie z.B. “getUnit” weiterverwendet werden. Das BTS2048 wird mit einer oder mehreren Kalibrierungen ausgeliefert. Diese dienen unterschiedlichen Messszenarien. Mit dieser Methode können Sie die im Eeprom abgelegten Kalibrierungen selektieren. Es existieren bis zu 52 Kalibriereinträge, von denen nicht jeder Speicherplatz mit einem Eintrag belegt sein muss.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*calibrationEntryNumber*	Pointer auf Integer Wert, enthält nach Rücksprung den selektierten Kalibriertabellenindex.

Rückgabe

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

5.3.3.6 GOMDBTS2048_readCalibrationEntryInfo()

```
int __stdcall GOMDBTS2048_readCalibrationEntryInfo (
            int handle,
```

```
            int calibrationEntryNumber,
            char * calibrationName )
```

Diese Methode liefert den im Eeprom definierten Namen des spezifizierten Kalibriertabelleneintrags zurück. Das BTS2048 wird mit einer oder mehreren Kalibrierungen ausgeliefert. Diese dienen unterschiedlichen Messszenarien. Mit dieser Methode können Sie die im Eeprom abgelegten Kalibrierungen selektieren. Es existieren bis zu 52 Kalibriereinträge, von denen nicht jeder Speicherplatz mit einem Eintrag belegt sein muss.
Für den Kalibriernamen muss vor Aufruf der Methode genügend Speicher (max. 256 Bytes) allokiert werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, der Kalibriertabellenindex, für den der Kalibriername ermittelt werden soll.
out	*calibrationName*	Nullterminierter String, der nach Rücksprung den Namen der Kalibrierung enthält.

Rückgabe

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

5.3.3.7 GOMDBTS2048_getMeasurementQuantity()

```
int __stdcall GOMDBTS2048_getMeasurementQuantity (
            int handle,
            int calibrationEntryNumber,
            char * quantity )
```

Diese Methode ermittelt für den spezifizierten Kalibriertabelleneintrag die Bezeichnung für die hinterlegte Messgröße zurück. Mögliche Antworten: “E”, “I” oder “Phi”.
Allokieren Sie entsprechend Speicher.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, der Kalibriertabellenindex, für den der Kalibriername ermittelt werden soll.
out	*quantity*	Nullterminierter String, “E”, “I”, “Phi”.

Rückgabe

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

5.3.3.8 GOMDBTS2048_isMeasurementQuantity()

```
int __stdcall GOMDBTS2048_isMeasurementQuantity (
            int handle,
```

```
            int calibrationEntryNumber,
            char * quantity,
            bool * isQuantity )
```

Mit dieser Methode kann überprüft werden, ob die Messgröße eines spezifizierten Kalibriertabellenindex einen bestimmten Wert beinhaltet. Es kann auf die Messgrößen “E”, “I” und “Phi” überprüft werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, der Kalibriertabellenindex, für den der Kalibriername ermittelt werden soll
in	*quantity*	Nullterminierter String, mögliche Werte: “E”, “I”, “Phi”
out	*isQuantity*	Pointer auf Boolean Wert: • true: Spezifizierte Kalibrierung dient der spezifizierten Messgröße • false: Spezifizierte Kalibrierung dient nicht der spezifizierten Messgröße

Rückgabe

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

5.3.3.9 GOMDBTS2048_getSelectedMeasurementQuantity()

```
int __stdcall GOMDBTS2048_getSelectedMeasurementQuantity (
            int handle,
            char * quantity )
```

Ermittelt die Messgrößenbezeichnung des aktuell selektierten Kalibriertabelleneintrags. Möglich Ergebnisse: „E”, “I” or “Phi”. Genügend Speicher muss vor Aufruf allokiert werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*quantity*	Nullterminierter String, enthält nach Rücksprung folgende mögliche Werte: “E”, “I”, “Phi”

Rückgabe

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

5.3.3.10 GOMDBTS2048_isMultiMeasurement()

```
int __stdcall GOMDBTS2048_isMultiMeasurement (
            int handle,
            int calibrationEntryNumber,
            bool * value )
```

Mit dieser Methode kann überprüft werden, ob eine Konfiguration (Kalibrierung) für eine Multimessung (aus mehreren Messbereichen zusammengesetzte Messung) definiert wurde.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, der Kalibriertabellenindex, für den „isMultiMeasurement“ ermittelt werden soll.
out	*value*	Pointer auf Boolean Wert: • true: Spezifizierte Kalibrierung ist für Multi-Measurement definiert • false: Spezifizierte Kalibrierung ist nicht für Multi-Measurement definiert

Rückgabe

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

5.3.3.11 GOMDBTS2048_isOORSLCorrectionMeasurement()

```
int __stdcall GOMDBTS2048_isOORSLCorrectionMeasurement (
             int handle,
             int calibrationEntryNumber,
             bool * value )
```

Mit dieser Methode kann überprüft werden, ob es sich bei einer Konfiguration (Kalibrierung) um eine eine OOR (Out of Range)-Streulichtkorrektur handelt. Dabei wird durch die Messung eines zusätlichen Filters das Streulicht im UV gemessen und abgezogen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, der Kalibriertabellenindex, für den „isOORSLCorrectionMeasurement“ ermittelt werden soll
out	*value*	Pointer auf Boolean Wert: • true: Spezifizierte Kalibrierung ist als Streulichtmessung definiert • false: Spezifizierte Kalibrierung ist nicht als Streulichtmessung definiert

Rückgabe

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

5.3.3.12 GOMDBTS2048_isSLMCorrectionMeasurement()

```
int __stdcall GOMDBTS2048_isSLMCorrectionMeasurement (
             int handle,
```

```
            int calibrationEntryNumber,
            bool * value )
```

Mit dieser Methode kann überprüft werden, ob es sich bei einer Konfiguration (Kalibrierung) um eine eine Messung mit SLM (Straylight Matrix) - Korrektur handelt. Dabei wird Anhand des gemessenen Spektrums das Streulicht berechnet und durch eine Matrixmultiplikation verrechnet.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, der Kalibriertabellenindex, für den „isSLMCorrectionMeasurement“ ermittelt werden soll
out	*value*	Pointer auf Boolean Wert: • true: SLM ist vorhanden • false: SLM ist nicht vorhanden

Rückgabe

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

5.3.3.13 GOMDBTS2048_setMeasurementMode()

```
int __stdcall GOMDBTS2048_setMeasurementMode (
            int handle,
            int measurementMode )
```

Diese Methode definiert, ob die Messung sofort ausgeführt wird, oder ob ein das Gerät auf ein Triggersignal warten soll. Zur Konfiguration des Triggers siehe entsprechende Befehle.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*measurementMode*	Integer Wert, der den gewünschten Modus enthält: • 0: Sofortige Ausführung • 1: Getriggerte Messung

Rückgabe

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

5.3.3.14 GOMDBTS2048_getMeasurementMode()

```
int __stdcall GOMDBTS2048_getMeasurementMode (
            int handle,
            int * measurementMode )
```

Diese Methode ermittelt den zuvor eingestellten Messmodus.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*measurementMode*	Pointer auf einen Integer Wert, der den gewünschten Modus enthält, • 0: Sofortige Ausführung • 1: Getriggerte Messung

Rückgabe

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

5.3.3.15 GOMDBTS2048_setSpectralIntegralSynch()

```
int __stdcall GOMDBTS2048_setSpectralIntegralSynch (
            int handle,
            bool value )
```

Mit dieser Methode kann bestimmt werden, ob die integrale Messung und die spektrale Messung synchron erfolgen soll. Andernfalls wird zuerst spektral und danach integral gemessen. Synchrone Messung ist besonders dann empfehlenswert wenn das Signal nur gepulst ist und nicht dauerhaft vorhanden ist. Außerdem kann durch die gleichzeitige Messung Zeit eingespart werden im Vergleich zu einer nacheinander durchgeführten Messung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true: Messung erfolgt synchron • false: Messung erfolgt nacheinander

Rückgabe

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

5.3.3.16 GOMDBTS2048_isSpectralIntegralSynch()

```
int __stdcall GOMDBTS2048_isSpectralIntegralSynch (
            int handle,
            bool * value )
```

Mit dieser Methode wird ermittelt, ob die synchrone Messung von integraler und spektraler Einheit eingeschaltet ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • true: Messung erfolgt synchron • false: Messung erfolgt nacheinander

Rückgabe

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

5.3.3.17 GOMDBTS2048_setDistance()

```
int __stdcall GOMDBTS2048_setDistance (
            int handle,
            double distance )
```

Wenn ein Kalibriereintrag für „Luminous Intensity“ oder „Radiant Intensity“ ausgewählt wurde, dann muss der Abstand zwischen dem Messgerät und dem Testobjekt definiert werden. Bei allen anderen Messgrößen muss der Abstand 1.0 betragen. Bei der Auswahl eines Kalibriereintrages ungleich der Messgröße „I“ wird automatisch der Wert 1.0 gesetzt. Die Einheit des Abstands ist [m].

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*distance*	Double Wert, enthält den Abstand in Meter [m].

Rückgabe

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

5.3.3.18 GOMDBTS2048_getDistance()

```
int __stdcall GOMDBTS2048_getDistance (
            int handle,
            double * distance )
```

Liefert den aktuell definierten Abstand zwischen dem Messgerät und dem Testobjekt. Die Einheit des Abstands ist [m].

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*distance*	Pointer auf Double Wert, enthält den Abstand in Meter [m].

Rückgabe

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

5.3.3.19 GOMDBTS2048_getFilterName()

```
int __stdcall GOMDBTS2048_getFilterName (
            int handle,
            int position,
            char * value )
```

Liefert den Namen des Filters zurück, der sich an einer bestimmten Position des Filterrads befindet.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*position*	Position am Filterrad.
out	*value*	Nullterminierter String, eindeutige Filterbezeichnung (maximal 39 Zeichen plus Nullterminator)

Rückgabe

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

5.3.3.20 GOMDBTS2048_getFilterNameforCalibration()

```
int __stdcall GOMDBTS2048_getFilterNameforCalibration (
            int handle,
            int calibrationEntryNumber,
            char * value )
```

Liefert den Namen des Filters zurück, der zur einer bestimmten Kalibrierung gehört. Wenn für eine Kalibrierung mehere Filter notwendig sind, werden sie mit einem Komma getrennt zurückgegben. Die Reihenfolge entspricht dabei der Reihenfolge, mit der sie angefahren werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Nummer des Kalibriereintrags.
out	*value*	Nullterminierter String, eindeutige Filterbezeichnung (maximal 319 Zeichen plus Nullterminator)

Rückgabe

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

5.4 Methoden für spektrale Messeinstellung

Messeinstellungen für die spektrale Messung.

Funktionen

- int __stdcall GOMDBTS2048_spectralSetEnabled (int handle, bool enabled)
- int __stdcall GOMDBTS2048_spectralIsEnabled (int handle, bool *enabled)
- int __stdcall GOMDBTS2048_spectralSetOffsetMode (int handle, int value)
- int __stdcall GOMDBTS2048_spectralGetOffsetMode (int handle, int *value)
- int __stdcall GOMDBTS2048_OORSLCorrectionSetMode (int handle, int value)
- int __stdcall GOMDBTS2048_OORSLCorrectionGetMode (int handle, int *value)
- int __stdcall GOMDBTS2048_spectralGetIntegrationTimeRangeInus (int handle, int *min, int *max)
- int __stdcall GOMDBTS2048_spectralSetIntegrationTimeInus (int handle, int timeInus)
- int __stdcall GOMDBTS2048_spectralGetIntegrationTimeInus (int handle, int *timeInus)
- int __stdcall GOMDBTS2048_spectralSetMeasurementTimeInUs (int handle, int value)
- int __stdcall GOMDBTS2048_spectralGetMeasurementTimeInUs (int handle, int *value)
- int __stdcall GOMDBTS2048_spectralSetDynamicTimeMode (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralGetDynamicTimeMode (int handle, bool *value)
- int __stdcall GOMDBTS2048_spectralSetMaxIntegrationTimeInUs (int handle, int value)
- int __stdcall GOMDBTS2048_spectralGetMaxIntegrationTimeInUs (int handle, int *value)
- int __stdcall GOMDBTS2048_spectralSetMaxMeasurementTimeInUs (int handle, int value)
- int __stdcall GOMDBTS2048_spectralGetMaxMeasurementTimeInUs (int handle, int *value)
- int __stdcall GOMDBTS2048_spectralSetNrOfScans (int handle, int nrOfScans)
- int __stdcall GOMDBTS2048_spectralGetNrOfScans (int handle, int *nrOfScans)
- int __stdcall GOMDBTS2048_setWavelengthRange (int handle, double L1, double L2, double dL)
- int __stdcall GOMDBTS2048_getWavelengthRange (int handle, double *L1, double *L2, double *dL)
- int __stdcall GOMDBTS2048_getMinValidWavelength (int handle, double *value)
- int __stdcall GOMDBTS2048_getMaxValidWavelength (int handle, double *value)

5.4.1 Ausführliche Beschreibung

5.4.2 C++ Aufrufbeispiel

Einen Offset-Modus auswählen, Offset-Messung durchführen und anschließend eine Messung.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);              //initialization
GOMDBTS2048_spectralSetOffsetMode(handle, 1);      //set offset mode;
       static offset
GOMDBTS2048_spectralMeasureOffset(handle);         // measure spectral
       offset
GOMDBTS2048_measure();                               //do a measurement
    //do something with results
GOMDBTS2048_releaseHandle(handle);                 //release handle
```

5.4.3 Dokumentation der Funktionen

5.4.3.1 GOMDBTS2048_spectralSetEnabled()

```
int __stdcall GOMDBTS2048_spectralSetEnabled (
             int handle,
             bool enabled )
```

Diese Methode aktiviert / deaktiviert das Spektrometer für die Messung. Falls das Spektrometer deaktiviert ist, wird der beim Anstoß der nächsten Gesamtmessung keine spektrale Messung durchgeführt. Der spektrale Messwert der zuletzt ausgeführten Messung bleibt dann erhalten. Per „default“ ist das Spektrometer nach Systemstart aktiviert. Wenn das Spektrometer deaktiviert ist, dann wird auch kein dynamischer a(z)-Korrekturfaktor berechnet. Dies bedeutet, dass der zuletzt berechnete a(z)-Faktor erhalten bleibt oder ein statischer Korrekturfaktor gesetzt werden sollte.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*enabled*	Boolean Wert: • true: Aktiviere Spektrometer zur Messung • false: Deaktiviere Spektrometer zur Messung

Rückgabe

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

5.4.3.2 GOMDBTS2048_spectralIsEnabled()

```
int __stdcall GOMDBTS2048_spectralIsEnabled (
            int handle,
            bool * enabled )
```

Diese Methode liest aus, ob das Spektrometer für die Messung aktiviert ist. Falls das Spektrometer deaktiviert ist, wird beim Anstoß der nächsten Gesamtmessung keine spektrale Messung durchgeführt. Die spektralen Werte der zuletzt ausgeführten Messung bleiben dann erhalten. Per „default“ ist das Spektrometer nach Systemstart aktiviert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*enabled*	Pointer auf einen Boolean-Wert: • true: Aktiviert • false: Deaktiviert

Rückgabe

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

5.4.3.3 GOMDBTS2048_spectralSetOffsetMode()

```
int __stdcall GOMDBTS2048_spectralSetOffsetMode (
            int handle,
            int value )
```

Diese Methode definiert den zur Messung verwendeten Offset Modus. Es gibt vier Unterschiedliche Modi: kein Offset, statisch, dynamisch und vorgemessen.
Kein Offset bedeutet, dass kein Offset vom gemessenen Signal abgezogen wird.
Statisch bedeutet, dass der Offset einmal explizit gemessen werden muss. Dies wird mit der Methode „spectral↩ MeasureOffset“ durchgeführt. Ab dem Moment wird dieser ermittelte Offset für die folgenden Messungen verwendet. Mit der Methode „spectralDeleteOffset“ wird der Offset auf 0 zurückgesetzt. Beim statischen Offset ist zu beachten, dass die spektrale Integrationszeit nach der Offsetmessung nicht mehr verändert werden darf, da sich der

Offset mit der Integrationszeit verändert. Nach einer Veränderung der spektralen Integrationszeit muss der Offset erneut gemessen werden.
Dynamisch bedeutet, dass der Offset bei jeder Messung neu ermittelt wird. Dazu wird der „Darkfilter“ automatisch gesetzt und der Offset gemessen. Für die tatsächliche Messung des Nutzsignals wird der Filter auf die zuvor eingestellte Position zurückgesetzt. Der Filter kann entweder explizit über die Methode „setFilterPosition“ eingestellt werden oder nach Setzen eines Kalibriereintrages wird der zu dem Kalibriereintrag zugehörige Filter verwendet.
Der Modus „vorgemessen“ ist für Messungen mit automatischer Integrationszeit verfügbar. Dabei wird der Offset für fest definierte Integrationszeiten vorgemessen. Dies muss manuell mit der Funktion spectralMeasure↩ PremeasuredOffset gemacht werden. Bei der Messung wird die Integrationszeit dann an eine der vordefinierten Zeiten angepasst.
Wenn mit dieser Methode die Funktionalität „kein Offset“ bzw. „dynamischer Offset“ gesetzt wird, wird die Methode „spectralDeleteOffset“ automatisch ausgeführt. Und somit der letzte gemessene Offset gelöscht.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert: • 0: Kein Offset • 1: Statischer Offset • 2: Dynamischer Offset • 3: Vorgemessener Offset

Rückgabe

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

5.4.3.4 GOMDBTS2048_spectralGetOffsetMode()

```
int __stdcall GOMDBTS2048_spectralGetOffsetMode (
              int handle,
              int * value )
```

Diese Methode ermittelt den zuvor gesetzten Offset Modus:

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Integer Wert: • 0: Kein Offset • 1: Statischer Offset • 2: Dynamischer Offset • 3: Vorgemessener Offset

Rückgabe

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

5.4.3.5 GOMDBTS2048_OORSLCorrectionSetMode()

```
int __stdcall GOMDBTS2048_OORSLCorrectionSetMode (
            int handle,
            int value )
```

Diese Methode steht nur bei OOR SLC (Out of Range Straylight Correction) Kalibriereinträgen zur Verfügung (ausschließlich bei BTS2048-UV Geräten).
Bei einer OOR SLC Kalibrierung wird das Streulicht mit einem eigenem Filter gemessen und dann von der Messung ohne Filter abgezogen. Für jeden Pixel kann mit der Methode OORSLCorrectionMeasureFactors() ein Faktor bestimmt werden wie sich dieser durch den Streulichabzug verändert. Wenn der OORSLCMode auf 1 gesetzt wird ist die Messung des Streulichts nicht mehr notwendig und die Messzeit reduziert sich.
Diese Methode kann nur verwendet werden, wenn sich die spektrale Verteile nicht verändert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert: • 0: Keine Vorgemessenen Faktoren • 1: Vorgemessene OOR SL-Correction Faktoren

Rückgabe

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

5.4.3.6 GOMDBTS2048_OORSLCorrectionGetMode()

```
int __stdcall GOMDBTS2048_OORSLCorrectionGetMode (
            int handle,
            int * value )
```

Diese Methode ermittelt den zuvor gesetzten OOR SL-Correction Modus:

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Integer Wert: • 0: Keine Vorgemessenen Faktoren • 1: Vorgemessene OOR SL-Correction Faktoren

Rückgabe

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

5.4.3.7 GOMDBTS2048_spectralGetIntegrationTimeRangeInus()

```
int __stdcall GOMDBTS2048_spectralGetIntegrationTimeRangeInus (
            int handle,
            int * min,
            int * max )
```

Liefert die kleinste und größte zulässige Integrationszeit für die spektrale Messeinheit des verbundenen BTS2048 in der Einheit [ms].

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*min*	Pointer auf Integer, enthält nach Rücksprung die kleinste zulässige Integrationszeit in Mikrosekunden.
out	*max*	Pointer auf Integer, enthält nach Rücksprung die größte zulässige Integrationszeit in Mikrosekunden.

Rückgabe

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

5.4.3.8 GOMDBTS2048_spectralSetIntegrationTimeInus()

```
int __stdcall GOMDBTS2048_spectralSetIntegrationTimeInus (
            int handle,
            int timeInus )
```

Mit dieser Methode kann die Integrationszeit des Spektrometers definiert werden. Die Integrationszeiten müssen in der Einheit µs an die Methode übergeben werden.
Wertebereich: 2 – 4000000 -> 2µs bis 4sec. Falls Sie ein Gerät mit Kühlung besitzen (Kühlung muss eingeschaltet sein), dann sind Werte bis 60000000 µs, also 60sec. zulässig. Wenn die Integrationszeit zu lange gewählt wurde, dann übersteuert das Spektrometer und die Messergebnisse können unbrauchbar sein.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*timeInus*	Integer Wert, die Integrationszeit in µSekunden.

Rückgabe

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

5.4.3.9 GOMDBTS2048_spectralGetIntegrationTimeInus()

```
int __stdcall GOMDBTS2048_spectralGetIntegrationTimeInus (
              int handle,
              int * timeInus )
```

Diese Methode liefert die für das Spektrometer zuletzt gesetzte Integrationszeit in der Einheit µs zurück.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*timeInus*	Pointer auf Integer; enthält nach Rücksprung die Integrationszeit in [µs].

Rückgabe

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

5.4.3.10 GOMDBTS2048_spectralSetMeasurementTimeInUs()

```
int __stdcall GOMDBTS2048_spectralSetMeasurementTimeInUs (
              int handle,
              int value )
```

Mit dieser Methode kann definiert werden, wie viel Zeit für die gesamte spektrale Messung (ohne Dunkelmessung). Für die meisten Kalibriereinträge sind Messzeit und Integrationszeit das gleiche und beide Funktionen (diese und "spectralSetIntegrationTimeInus()") können auf die gleiche Wiese benutzt werden. Bei BTS2048-UV Geräten gibt es bestimmte Kalibriereinträge, denen eine Kombinationsmessung mit verschiedenen Filterpositionen zugrunde liegt. In diesem Fall definiert die Integrationzeit die Dauer für eine FItermessung, die Messzeit hingegen die gesamte Messdauer.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert, die Messzeit in µSekunden.

Rückgabe

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

5.4.3.11 GOMDBTS2048_spectralGetMeasurementTimeInUs()

```
int __stdcall GOMDBTS2048_spectralGetMeasurementTimeInUs (
              int handle,
              int * value )
```

Diese Methode liefert die für das Spektrometer zuletzt gesetzte Messzeit (für Multimessungen) in der Einheit µs zurück.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer; enthält nach Rücksprung die Integrationszeit in [µs]

Rückgabe

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

5.4.3.12 GOMDBTS2048_spectralSetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_spectralSetDynamicTimeMode (
             int handle,
             bool value )
```

Diese Methode legt fest, ob zu jeder Messung die dynamische Ermittlung der Integrationszeit für die spektrale Maßeinheit durchgeführt werden soll.
Falls die dynamische Zeitermittlung eingeschaltet ist, führt das Messgerät vor der eigentlichen Messung eine Premessung durch. Die tatsächliche verwendete Integrationszeit kann dann nach der Messung mit „spectralGet↩ IntegrationTimeInus“ abgefragt werden. Dynamische Ermittlung der Integrationszeit ist nicht kompatibel mit dem „statischen dark modus“. Wenn die Dynamik eingeschaltet wird, wird der dark mode automatisch auf „dynamisch“ geändert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • false: Dynamische Integrationszeitermittlung deaktiviert • true: Dynamische Integrationszeitermittlung aktiviert

Rückgabe

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

5.4.3.13 GOMDBTS2048_spectralGetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_spectralGetDynamicTimeMode (
             int handle,
             bool * value )
```

Diese Methode liefert die Information zurück, ob zu jeder Messung die dynamische Ermittlung der Integrationszeit für die spektrale Messeinheit durchgeführt werden soll.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean: • false: Dynamische Integrationszeitermittlung deaktiviert • true: Dynamische Integrationszeitermittlung aktiviert

Rückgabe

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

5.4.3.14 GOMDBTS2048_spectralSetMaxIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_spectralSetMaxIntegrationTimeInUs (
            int handle,
            int value )
```

Mit dieser Methode wird die maximale Integrationszeit festgelegt. Wenn die dynamische Ermittelung der Integrationzeit angeschalten ist, ist sichergestellt, dass die maximale Integrationszeit nicht überschritten wird.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Doublewert, max. Integrationzeit in µs.

Rückgabe

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

5.4.3.15 GOMDBTS2048_spectralGetMaxIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_spectralGetMaxIntegrationTimeInUs (
            int handle,
            int * value )
```

Liefert den aktuellen Wert, auf den die maximale Integrationszeit gesetzt ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Doublewert, enthält die maximale Integrationszeit in µs.

Rückgabe

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

5.4.3.16 GOMDBTS2048_spectralSetMaxMeasurementTimeInUs()

```
int __stdcall GOMDBTS2048_spectralSetMaxMeasurementTimeInUs (
            int handle,
            int value )
```

Mit dieser Methode wird die maximale Messzeit festgelegt. Wenn die dynamische Ermittelung der Integrationzeit angeschalten ist, ist sichergestellt, dass die maximale Messzeit nicht überschritten wird. Für den Unterschied zwischen Integrationszeit und Messzeit betrachten SIe die Funktion spectralSetMeasurementTimeInUs

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Doublewert, max. Messzeit in µs.

Rückgabe

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

5.4.3.17 GOMDBTS2048_spectralGetMaxMeasurementTimeInUs()

```
int __stdcall GOMDBTS2048_spectralGetMaxMeasurementTimeInUs (
            int handle,
            int * value )
```

Liefert den aktuellen Wert, auf den die maximale Messzeit gesetzt ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Doublewert, enthält die maximale Messzeit in µs.

Rückgabe

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

5.4.3.18 GOMDBTS2048_spectralSetNrOfScans()

```
int __stdcall GOMDBTS2048_spectralSetNrOfScans (
            int handle,
            int nrOfScans )
```

Diese Methode definiert die Anzahl der Mittelungen Ihrer spektralen Messung. Zu diesem Zweck wird Ihr Spektrum entsprechend oft spektral vermessen und die Ergebnisse gemittelt. Dies verbessert das Ergebnis Ihrer Messung hinsichtlich Signal zu Rauschverhältnis, allerdings erhöht sich die Messzeit dementsprechend. Diese Einstellung muss vor dem Aufruf der Methode „measure“ erfolgen. Defaultwert nach Initialisierung ist „1“.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*nrOfScans*	Integer Wert, enthält die Anzahl der spektralen Messungen über die gemittelt werden soll pro Messvorgang.

Rückgabe

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

5.4.3.19 GOMDBTS2048_spectralGetNrOfScans()

```
int __stdcall GOMDBTS2048_spectralGetNrOfScans (
            int handle,
            int * nrOfScans )
```

Diese Methode liefert die für das Spektrometer zuvor gesetzte Anzahl an spektralen Messungen über die gemittelt werden soll.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*nrOfScans*	Pointer auf Integerwert; enthält die Anzahl der Mittelungen

Rückgabe

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

5.4.3.20 GOMDBTS2048_setWavelengthRange()

```
int __stdcall GOMDBTS2048_setWavelengthRange (
            int handle,
            double L1,
            double L2,
            double dL )
```

Mit dieser Methode definieren Sie die eingrenzenden Wellenlängenbereiche für spätere Aufrufe von “spectral↩ GetCountsWavelength” oder “spectralGetSpectrumCalibratedWavelength”. Der Wellenlängenbereich hat ebenso Einfluss auf die Berechnung der Halbwertsbreite, die mittels „getFWHM“ erhalten werden kann.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*L1*	Doublewert, die minimale Wellenlänge in [nm]
in	*L2*	Doublewert, die maximale Wellenlänge in [nm]
in	*dL*	Doublewert, Schrittweite in [nm]

Rückgabe

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

5.4.3.21 GOMDBTS2048_getWavelengthRange()

```
int __stdcall GOMDBTS2048_getWavelengthRange (
            int handle,
            double * L1,
            double * L2,
            double * dL )
```

Liefert den zuvor festgelegten Wellenlängenbereich, der in anderen Methoden wie z.B. “spectralGetCounts↩ Wavelength” oder “spectralGetSpectrumCalibratedWavelength” Verwendung findet.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*L1*	Pointer auf Doublewert, enthält die minimale Wellenlänge in [nm]
out	*L2*	Pointer auf Doublewert, enthält die maximale Wellenlänge in [nm]
out	*dL*	Pointer auf Doublewert, enthält die Schrittweite in [nm]

Rückgabe

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

5.4.3.22 GOMDBTS2048_getMinValidWavelength()

```
int __stdcall GOMDBTS2048_getMinValidWavelength (
            int handle,
            double * value )
```

Liefert die für den aktuell ausgewählten Kalibriereintrag minimal zulässige Wellenlänge.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Doublewert, enthält die minimale zulässige Wellenlänge in [nm]

Rückgabe

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

5.4.3.23 GOMDBTS2048_getMaxValidWavelength()

```
int __stdcall GOMDBTS2048_getMaxValidWavelength (
              int handle,
              double * value )
```

Liefert die für den aktuell ausgewählten Kalibriereintrag maximal zulässige Wellenlänge.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Doublewert, enthält die maximale zulässige Wellenlänge in [nm]

Rückgabe

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

5.5 Spektrale Korrekturmethoden und Filter

Funktionen

- int __stdcall GOMDBTS2048_spectralSetScaleWithIntegralMode (int handle, int value)
- int __stdcall GOMDBTS2048_spectralGetScaleWithIntegralMode (int handle, int *value)
- int __stdcall GOMDBTS2048_spectralSetScaleWithVLambda (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralIsScaleWithVLambda (int handle, bool *value)
- int __stdcall GOMDBTS2048_spectralSetPixelLinearization (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralIsPixelLinearization (int handle, bool *value)
- int __stdcall GOMDBTS2048_spectralSetBandwidthCorrection (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralIsBandwidthCorrection (int handle, bool *value)
- int __stdcall GOMDBTS2048_spectralSetSavitzkyGolayFilter (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralIsSavitzkyGolayFilter (int handle, bool *value)
- int __stdcall GOMDBTS2048_spectralSetNoiseReduction (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralIsNoiseReduction (int handle, bool *value)
- int __stdcall GOMDBTS2048_spectralSetDarkThreshold (int handle, int value)
- int __stdcall GOMDBTS2048_spectralGetDarkThreshold (int handle, int *value)
- int __stdcall GOMDBTS2048_spectralSetObserver10Degree (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralIsObserver10Degree (int handle, bool *value)
- int __stdcall GOMDBTS2048_spectralSetAdvancedNoiseReduction (int handle, bool value)
- int __stdcall GOMDBTS2048_spectralIsAdvancedNoiseReduction (int handle, bool *value)
- int __stdcall GOMDBTS2048_setPreciseCountCalculation (int handle, bool value)
- int __stdcall GOMDBTS2048_getPreciseCountCalculation (int handle, bool *value)

5.5.1 Ausführliche Beschreibung

5.5.2 Dokumentation der Funktionen

5.5.2.1 GOMDBTS2048_spectralSetScaleWithIntegralMode()

```
int __stdcall GOMDBTS2048_spectralSetScaleWithIntegralMode (
             int handle,
             int value )
```

Mit dieser Methode wird der Skalierungs Modus des Spektrums gesetzt. Falls das scaling aktiv ist, wird die spektrale Funktion so skaliert, dass der radiometrische Wert mit dem intrgealen Detektors zusammen passt. Es gibt drei Modi:

- 0: Aus -> Skalierung ausgeschaltet
- 1: Immer an -> Skalierung immer angeschaltet
- 2: Automatisch -> Skalierung angeschaltet falls empfohlen. Dies bedeutet das Spektrum wird nur skaliert, wenn der integrale Detektor genug signal hat, der AZ-Modus auf dynamisch oder automatisch steht und der skalierungs-Faktor das Siganl nicht mehr als 20% ändert.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integraler Wert: • 0: Aus • 1: Immer an • 2: Automatisch

Rückgabe

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

5.5.2.2 GOMDBTS2048_spectralGetScaleWithIntegralMode()

```
int __stdcall GOMDBTS2048_spectralGetScaleWithIntegralMode (
            int handle,
            int * value )
```

Ruft den aktuell eingestellten skalierungs Modus der spaktralen Funktion ab

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integraler Wert: • 0: Aus • 1: Immer an • 2: Automatisch

Rückgabe

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

5.5.2.3 GOMDBTS2048_spectralSetScaleWithVLambda()

```
int __stdcall GOMDBTS2048_spectralSetScaleWithVLambda (
            int handle,
            bool value )
```

Mit dieser Methode wird die Skalierung ein- bzw. ausgeschaltet. Wenn die Skalierung eingeschaltet ist, werden die spektralen Daten mit Hilfe des integralen Sensors absolut skaliert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true: Skalierung angeschaltet • false: Skalierung ausgeschaltet

Rückgabe

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

5.5.2.4 GOMDBTS2048_spectralIsScaleWithVLambda()

```
int __stdcall GOMDBTS2048_spectralIsScaleWithVLambda (
            int handle,
            bool * value )
```

Ermittelt, ob die Skalierung eingeschaltet ist.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • true: Skalierung angeschaltet • false: Skalierung ausgeschaltet

Rückgabe

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

5.5.2.5 GOMDBTS2048_spectralSetPixelLinearization()

```
int __stdcall GOMDBTS2048_spectralSetPixelLinearization (
            int handle,
            bool value )
```

Mit dieser Methode wird die Pixel-Linearisierung ein- bzw. ausgeschaltet. Sie bewirkt die Linearisierung des spektralen Sensors.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true: Linearisierung angeschaltet • false: Linearisierung ausgeschaltet

Rückgabe

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

5.5.2.6 GOMDBTS2048_spectralIsPixelLinearization()

```
int __stdcall GOMDBTS2048_spectralIsPixelLinearization (
            int handle,
            bool * value )
```

Ermittelt, ob die Pixel Linearisierung eingeschaltet ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • true: Linearisierung angeschaltet • false: Linearisierung ausgeschaltet

Rückgabe

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

5.5.2.7 GOMDBTS2048_spectralSetBandwidthCorrection()

```
int __stdcall GOMDBTS2048_spectralSetBandwidthCorrection (
            int handle,
            bool value )
```

Mit dieser Methode wird die Bandbreitenkorrektur ein- bzw. ausgeschaltet. Die Bandbreitenkorrektur basiert auf einer angepassten Methode von Woolliams welche laut CIE TC2.51 empfohlen wird.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true: Korrektur aktiv • false: Korrektur nicht aktiv

Rückgabe

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

5.5.2.8 GOMDBTS2048_spectralIsBandwidthCorrection()

```
int __stdcall GOMDBTS2048_spectralIsBandwidthCorrection (
            int handle,
            bool * value )
```

Ermittelt, ob die Bandbreitenkorrektur aktiviert ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Parameter

out	*value*	Pointer auf Boolean Wert: • true: Korrektur aktiv • false: Korrektur nicht aktiv

Rückgabe

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

5.5.2.9 GOMDBTS2048_spectralSetSavitzkyGolayFilter()

```
int __stdcall GOMDBTS2048_spectralSetSavitzkyGolayFilter (
            int handle,
            bool value )
```

Mit dieser Methode wird Rauschreduzierung des spektralen Sensors nach Savitzky-Golay aktiviert bzw. deaktiviert. Dieser Algorithmus kann nicht zugleich mit dem anderen Rauschreduktionsalgorithmus („spectralSetNoise↩ Reduction“) angewendet werden. Bei Aktivierung des einen Algorithmus wird der andere automatisch deaktiviert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true: Rauschreduzierung aktiv • false: Rauschreduzierung nicht aktiv

Rückgabe

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

5.5.2.10 GOMDBTS2048_spectralIsSavitzkyGolayFilter()

```
int __stdcall GOMDBTS2048_spectralIsSavitzkyGolayFilter (
            int handle,
            bool * value )
```

Ermittelt, ob die Rauschreduzierung nach Savitzky-Golay für den spektralen Sensor aktiviert ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • true: Rauschreduzierung aktiv • false: Rauschreduzierung nicht aktiv

Rückgabe

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

5.5.2.11 GOMDBTS2048_spectralSetNoiseReduction()

```
int __stdcall GOMDBTS2048_spectralSetNoiseReduction (
            int handle,
            bool value )
```

Mit dieser Methode wird Rauschreduzierung des spektralen Sensors aktiviert bzw. deaktiviert. Bei dieser Methode werden +-2 benachbarte Pixel gemittelt. Dies ist legitim, da diese Pixel noch innerhalb der Bandbreite des Geräts liegen. Dieser Algorithmus ist bei verrauschten Signalen empfehlenswert, da mit einer einfachen Messung quasi eine 5 fache Mittelung stattfindet. Besonders empfehlenswert ist dies bei breitbandigen Lichtquellen. Bei Linienlampen oder Lasern führt dieser Algorithmus zu einer Bandbreitenverbreiterung und ist gegebenenfalls nicht die optimale Wahl. Dieser Algorithmus kann nicht zugleich mit dem anderen Rauschreduktionsalgorithmus („spectral↩ SetSavitzkyGolayFilter“) angewendet werden. Bei Aktivierung des einen Algorithmus wird der andere automatisch deaktiviert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true -$>$ Rauschreduzierung aktiv • false -$>$ Rauschreduzierung nicht aktiv

Rückgabe

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

5.5.2.12 GOMDBTS2048_spectralIsNoiseReduction()

```
int __stdcall GOMDBTS2048_spectralIsNoiseReduction (
            int handle,
            bool * value )
```

Ermittelt, ob die Rauschreduzierung des spektralen Sensors aktiviert ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean-Wert: • true: Rauschreduzierung aktiv • false: Rauschreduzierung nicht aktiv

Rückgabe

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

5.5.2.13 GOMDBTS2048_spectralSetDarkThreshold()

```
int __stdcall GOMDBTS2048_spectralSetDarkThreshold (
            int handle,
            int value )
```

Der Dunkelschwellwert definiert die Anzahl an Counts, die minimal vorhanden sein muss, damit das Signal an dem jeweiligen Pixel ausgewertet und verarbeitet wird. Wertebereich: 0 - 65535

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert; Anzahl der Counts die als Dunkelschwellwert überschritten werden muss. Wertebereich: 0 - 65535

Rückgabe

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

5.5.2.14 GOMDBTS2048_spectralGetDarkThreshold()

```
int __stdcall GOMDBTS2048_spectralGetDarkThreshold (
            int handle,
            int * value )
```

Ermittelt die Anzahl an Counts, die als Dunkelschwelle definiert wurden. Möglicher Wertebereich: 0 - 65535.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer Wert; Anzahl der Counts die als Dunkelschwelle definiert wurden.

Rückgabe

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

5.5.2.15 GOMDBTS2048_spectralSetObserver10Degree()

```
int __stdcall GOMDBTS2048_spectralSetObserver10Degree (
            int handle,
            bool value )
```

Diese Methode legt fest welcher CIE Normalbeobachter für die Farbberechnung verwendet werden soll.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • false: CIE 1931 Normalbeobachter mit 2° Blickfeld • true: CIE 1964 Normalbeobachter mit 10° Blickfeld

Rückgabe

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

5.5.2.16 GOMDBTS2048_spectralIsObserver10Degree()

```
int __stdcall GOMDBTS2048_spectralIsObserver10Degree (
            int handle,
            bool * value )
```

Diese Methode gibt zurück, welcher CIE Normalbeobachter für die Farbberechnung verwendet wird.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • false: CIE 1931 Normalbeobachter mit 2° Blickfeld • true: CIE 1964 Normalbeobachter mit 10° Blickfeld

Rückgabe

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

5.5.2.17 GOMDBTS2048_spectralSetAdvancedNoiseReduction()

```
int __stdcall GOMDBTS2048_spectralSetAdvancedNoiseReduction (
            int handle,
            bool value )
```

Diese Methode schaltet die verbesserte Rauschunterdrückung ein. Die Rauschunterdrückung ist ein speziell entwickelter Filter, der das Spektrum dynamisch glättet und Rauschen unterdrückt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Parameter

in	*value*	Boolean Wert: • false: Deaktivieren • true: Aktivieren

Rückgabe

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

5.5.2.18 GOMDBTS2048_spectralIsAdvancedNoiseReduction()

```
int __stdcall GOMDBTS2048_spectralIsAdvancedNoiseReduction (
            int handle,
            bool * value )
```

Diese Methode gibt zurück, ob die verbesserte Rauschunterdrückung aktiviert ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • false: Nicht aktiv • true: Aktiv

Rückgabe

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

5.5.2.19 GOMDBTS2048_setPreciseCountCalculation()

```
int __stdcall GOMDBTS2048_setPreciseCountCalculation (
            int handle,
            bool value )
```

Diese Methode legt fest ob für die Berechnung der Counts Integer-Variablen oder Floating-Point-Variablen verwendet werden sollen. Die floating point Variante ist genauer, dafür kann sich die Berechnung aber auf die performance auswirken. Besonders wenn die Anzahl der Mittelungen „NrOfScans“ auf größer 1 gesetzt wird, ist dies Verwendung der PreciseCountCalculation() zu empfehlen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • false: Integer Genauigkeit • true: Hohe floating point Genauigkeit

Rückgabe

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

5.5.2.20 GOMDBTS2048_getPreciseCountCalculation()

```
int __stdcall GOMDBTS2048_getPreciseCountCalculation (
            int handle,
            bool * value )
```

Diese Methode gibt zurück, welche Genauigkeit für die Berechnung der Counts verwendet wird (integer oder floating point) .

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Boolean Wert: • false: Integer Genauigkeit • true: Hohe floating point Genauigkeit

Rückgabe

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

5.6 Integrale Messeinstellungen

Funktionen

- int __stdcall GOMDBTS2048_integralSetEnabled (int handle, bool enabled)
- int __stdcall GOMDBTS2048_integralIsEnabled (int handle, bool *enabled)
- int __stdcall GOMDBTS2048_integralGetIntegrationTimeRangeInMs (int handle, int *min, int *max)
- int __stdcall GOMDBTS2048_integralSetIntegrationTimeInUs (int handle, int range, int value)
- int __stdcall GOMDBTS2048_integralGetIntegrationTimeInUs (int handle, int range, int *time)
- int __stdcall GOMDBTS2048_setIntegralRange (int handle, int value)
- int __stdcall GOMDBTS2048_getIntegralRange (int handle, int *value)
- int __stdcall GOMDBTS2048_integralSetRangeWaitTimeInMs (int handle, int rangeArea, int value)
- int __stdcall GOMDBTS2048_integralGetRangeWaitTimeInMs (int handle, int rangeArea, int *value)
- int __stdcall GOMDBTS2048_integralSetAzMode (int handle, int mode)
- int __stdcall GOMDBTS2048_integralGetAzMode (int handle, int *mode)
- int __stdcall GOMDBTS2048_integralSetAzSpecific (int handle, double az)
- int __stdcall GOMDBTS2048_integralGetAzSpecific (int handle, double *az)

5.6.1 Ausführliche Beschreibung

5.6.2 C++ Aufrufbeispiel

Eine Integrationszeit von 5000 us setzen.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                          //initialization
GOMDBTS2048_integralSetIntegrationTimeInUs(handle, 1, 5000);  //
      sets integration time, 5000 us
GOMDBTS2048_releaseHandle(handle);                             //release handle
```

5.6.3 Dokumentation der Funktionen

5.6.3.1 GOMDBTS2048_integralSetEnabled()

```
int __stdcall GOMDBTS2048_integralSetEnabled (
              int handle,
              bool enabled )
```

Diese Methode aktiviert / deaktiviert den integralen Sensor für die Messung. Falls der integrale Sensor deaktiviert ist, wird der beim Anstoß der nächsten Gesamtmessung keine integrale Messung durchgeführt. Der integrale Messwert der zuletzt ausgeführten Messung bleibt dann erhalten. Per „default“ ist der integrale Sensor nach Systemstart aktiviert.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*enabled*	Boolean Wert: • true: Aktiviere integralen Sensor zur Messung • false: Deaktiviere integralen Sensor zur Messung

Rückgabe

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

5.6.3.2 GOMDBTS2048_integralIsEnabled()

```
int __stdcall GOMDBTS2048_integralIsEnabled (
            int handle,
            bool * enabled )
```

Diese Methode liest aus, ob der integrale Sensor für die Messung aktiviert ist. Falls der integrale Sensor deaktiviert ist, wird beim Anstoß der nächsten Gesamtmessung keine integrale Messung durchgeführt. Der integrale Messwert der zuletzt ausgeführten Messung bleibt dann erhalten. Per „default“ ist der integrale Sensor nach Systemstart aktiviert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*enabled*	Pointer auf einen Boolean Wert: • true: Aktiviert • false: Deaktiviert

Rückgabe

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

5.6.3.3 GOMDBTS2048_integralGetIntegrationTimeRangeInMs()

```
int __stdcall GOMDBTS2048_integralGetIntegrationTimeRangeInMs (
            int handle,
            int * min,
            int * max )
```

Liefert die kleinste und größte zulässige Integrationszeit für die integrale Messeinheit des verbundenen + BTS2048 in der Einheit [ms]

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*min*	Pointer auf Integer, enthält nach Rücksprung die kleinste zulässige Integrationszeit in Millisekunden.
out	*max*	Pointer auf Integer, enthält nach Rücksprung die größte zulässige Integrationszeit in Millisekunden.

Rückgabe

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

5.6.3.4 GOMDBTS2048_integralSetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_integralSetIntegrationTimeInUs (
            int handle,
            int range,
            int value )
```

Mit dieser Methode kann die Integrationszeit für den integralen Sensor definiert werden. Die Integrationszeiten müssen in der Einheit [us] an die Methode übergeben werden.
Wertebereich bei Flicker-Geräten mit schnellem integralen Verstärker: 20 – 2000000 entspricht 20 us bis 2 s in 20us-Schritten.
Wertebereich bei normalen BTS2048 Geräten: 1000 – 2000000 -> 1 ms bis 2 s.
Es gibt jeweils eine Integrationszeit für den Messbereich 0 – 5 (Parameter range = 0) sowie für den Messbereich 6 – 8 (Parameter range = 1).
Nach dem Setzen einer neuen Integrationszeit wird automatisch eine „Zeromessung“ durchgeführt, die abhängig vom gewählten Messbereich zusätzliche Verarbeitungszeit kostet. Diese setzt sich zusammen aus der tatsächlichen Integrationszeit für die „Zeromessung“ sowie eine Wartezeit vor Beginn der Messung (derzeit 500ms für Messbereich 0-5 und 1000ms für Messbereich 6 – 8).
Falls die neue gesetzte Zeit identisch ist mit der bereits aktiven Integrationszeit entfällt die Zeromessung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*range*	Integer Wert • 0: Für Messbereich 0 – 5 • 1: Für Messbereich 6 - 8
in	*value*	Integer Wert, die Integrationszeit in Mikrosekunden [us].

Rückgabe

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

5.6.3.5 GOMDBTS2048_integralGetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_integralGetIntegrationTimeInUs (
            int handle,
            int range,
            int * time )
```

Diese Methode liefert die für den integralen Sensor zuletzt gesetzte Integrationszeit bzgl. des gewünschten Range-Bereiches in der Einheit [us].

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*range*	Integer Wert: • 0: Für Messbereich 0 – 5 • 1: Für Messbereich 6 - 8
out	*time*	Pointer auf Integer; die Integrationszeit in Mikrosekunden [us].

Rückgabe

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

5.6.3.6 GOMDBTS2048_setIntegralRange()

```
int __stdcall GOMDBTS2048_setIntegralRange (
            int handle,
            int value )
```

Definiert den Messbereich, in dem sich das zu messende Signal befindet.

Range	Range max.	rise time normal version (10 - 90%)	rise time “flicker” version (10-90%)	gain error ± offset error(at 20 °C)
0	±20µA	1ms	50us	0.2% ±0.2µA
1	±4.3µA	1ms	50us	0.2% ±0.004µA
2	±920nA	1ms	50us	0.2% ±0.001nA
3	±200nA	2.5ms	65us	0.2% ±0.2nA
4	±43nA	2.5ms	65us	0.2% ±0.04nA
5	±9.2nA	2.5ms	65us	0.5% ±10pA
6	±2.0nA	5ms	1.5ms	0.5% ±2pA
7	±430pA	5ms	1.5ms	0.5% ±2pA
8	±92pA	5ms	1.5ms	0.5% ±2pA

Range = -1 bedeutet „auto-ranging“. Das Gerät sucht sich selbständig den optimalen zum Signal passenden Messbereich.
Das automatische Umschalten kann bei getriggerten Messungen zu undefinierten Ergebnissen führen und geht zusätzlich zu Lasten der Performance.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integerwert; enthält den Messbereich: • -1: Automatische Messbereichsermittlung • 0 – 8: Spezifische Messbereiche (0 = unempfindlich, 8 = empfindlich)

Rückgabe

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

5.6.3.7 GOMDBTS2048_getIntegralRange()

```
int __stdcall GOMDBTS2048_getIntegralRange (
            int handle,
            int * value )
```

Ermittelt den aktuell eingestellten Messbereich.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integerwert; enthält den aktuell gesetzten Messbereich: • -1: auto ranging • 0: unempfindlich • . . . • 8: empfindlich

Rückgabe

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

5.6.3.8 GOMDBTS2048_integralSetRangeWaitTimeInMs()

```
int __stdcall GOMDBTS2048_integralSetRangeWaitTimeInMs (
            int handle,
            int rangeArea,
            int value )
```

Mit dieser Methode kann die Wartezeit für das Umschalten des integralen Range-Bereiches gesetzt werden. Die Wartezeiten müssen in der Einheit [ms] an die Methode übergeben werden. Im Range-Bereich 0 sind Wartezeiten von 5 ms - 20 ms möglich, im Range-Bereich 1 sind Wartezeiten zwischen 20 ms und 200 ms zulässig.
Defaultwerte:
rangeArea 0: value = 20 ms
rangeArea 1: value = 200 ms
Die Defaultwerte sollten nur überschrieben werden, wenn aus perfomrmance-technischen Gründen keine längeren Wartezeiten möglich sind.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*rangeArea*	Integer Wert: • 0: Für Messbereich 0 – 5 • 1: Für Messbereich 6 – 8
in	*value*	die Wartezeit in Millisekunden.

Rückgabe

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

5.6.3.9 GOMDBTS2048_integralGetRangeWaitTimeInMs()

```
int __stdcall GOMDBTS2048_integralGetRangeWaitTimeInMs (
            int handle,
            int rangeArea,
            int * value )
```

Diese Methode liefert die für den integralen Sensor zuletzt gesetzte Wartezeit beim Umschalten des integralen Range-Bereiches in der Einheit [ms].

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*rangeArea*	Integer Wert: • 0: Für Messbereich 0 – 5 • 1: Für Messbereich 6 - 8
out	*value*	Pointer auf Integer; die Wartezeit in Millisekunden.

Rückgabe

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

5.6.3.10 GOMDBTS2048_integralSetAzMode()

```
int __stdcall GOMDBTS2048_integralSetAzMode (
            int handle,
            int mode )
```

Mit dieser Methode wird die a(z) Strategie definiert. Die a(z)-Korrektur dient der spektralen Anpassung des integralen Sensors. Es existieren folgende Möglichkeiten:

- mode = 0: keine spektrale Anpassung
- mode = 1: spektrale Anpassung mit einem statischen Korrekturfaktor der mit der Methode “integralSetAz↩ Specific” definiert werden kann.
- mode = 2: dynamische spektrale Anpassung. Nach jeder Messung wird ein neuer a(z)-Korrekturfaktor aus den spektralen Daten berechnet; dazu muss die spektrale Messung aktiviert sein.
- mode = 3: automatische spektrale Anpassung. Ein neuer a(z)-Korrekturfaktor wird wie im Modus 2 berechnet, aber nur wenn das Signal im relevanten Spektralbereich ausreichend ist. Andernfalls ist der a(z)-↩ Korrekturfaktor gleich 1; spektrale Messung muss aktiviert sein.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*mode*	Integer Wert, der zu setzende a(z)-Modus

Rückgabe

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

5.6.3.11 GOMDBTS2048_integralGetAzMode()

```
int __stdcall GOMDBTS2048_integralGetAzMode (
              int handle,
              int * mode )
```

Die a(z) Korrektur Diese Methode liefert den zuvor gesetzten a(z) Korrekturmodus. Es gibt folgende Modi:

- mode = 0: keine a(z) Korrektur
- mode = 1: Korrektur mit einem statischen Faktor, der durch einen Aufruf von “integralSetAzSpecific” gesetzt wurde.
- mode = 2: Korrektur mit einem dynamisch berechneten a(z) Korrekturfaktor, auf Basis einer zuvor durchgeführten spektralen Messung.
- mode = 2: Korrektur mit einem automatisch berechneten a(z) Korrekturfaktor, auf Basis einer zuvor durchgeführten spektralen Messung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*mode*	Pointer auf einen Integerwert; enthält den zuvor gesetzten a(z) mode

Rückgabe

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

5.6.3.12 GOMDBTS2048_integralSetAzSpecific()

```
int __stdcall GOMDBTS2048_integralSetAzSpecific (
              int handle,
              double az )
```

Definiert einen statischen a(z)-Korrekturfaktor. Der Korrekturfaktor dient der spektralen Anpassung des integralen Sensors. Dieser Wert wird nur verwendet, wenn gleichzeitig der a(z)-Modus auf den Wert „1“ gesetzt wird.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*az*	Double Wert, der den statischen a(z)-Korrekturfaktor repräsentiert.

Rückgabe

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

5.6.3.13 GOMDBTS2048_integralGetAzSpecific()

```
int __stdcall GOMDBTS2048_integralGetAzSpecific (
            int handle,
            double * az )
```

Ermittelt den zuvor definierten statischen a(z)-Korrekturfaktor. Der Korrekturfaktor dient der spektralen Anpassung des integralen Sensors. Dieser Wert wird nur verwendet, wenn gleichzeitig der a(z)-Modus auf den Wert „1“ gesetzt ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*az*	Pointer auf einen Double Wert; enthält nach Rücksprung den statischen a(z)-Korrekturfaktur.

Rückgabe

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

5.7 Farbberechnungsmethoden

Funktionen

- int __stdcall GOMDBTS2048_setColorCalculation (int handle, bool value)
- int __stdcall GOMDBTS2048_isColorCalculation (int handle, bool *value)
- int __stdcall GOMDBTS2048_setColorCalculationMode (int handle, int value)
- int __stdcall GOMDBTS2048_getColorCalculationMode (int handle, int *value)
- int __stdcall GOMDBTS2048_setColorCalculationOptimizationMode (int handle, int value)
- int __stdcall GOMDBTS2048_getColorCalculationOptimizationMode (int handle, int *value)
- int __stdcall GOMDBTS2048_setCTLimitCheckActive (int handle, bool value)
- int __stdcall GOMDBTS2048_isCTLimitCheckActive (int handle, bool *value)
- int __stdcall GOMDBTS2048_setDeltaUVLimit (int handle, double limit)
- int __stdcall GOMDBTS2048_getDeltaUVLimit (int handle, double *limit)

5.7.1 Ausführliche Beschreibung

5.7.2 C++ Aufrufbeispiel

Farbtemperatur in Abhängigkeit des uv-Limits berechnen.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                  //initialization
GOMDBTS2048_setUVDeltaLimit(handle, 0.05);              //set of uv-limit on a value
GOMDBTS2048_setCTLimitCheckActive(handle, true);       //set the limit on
       true; now it´s possible to calculate the CCT according to the uv-limit
GOMDBTS2048_releaseHandle(handle);                     //release handle
```

5.7.3 Dokumentation der Funktionen

5.7.3.1 GOMDBTS2048_setColorCalculation()

```
int __stdcall GOMDBTS2048_setColorCalculation (
              int handle,
              bool value )
```

Nur bei aktivierter „color calculation“, werden nach der spektralen Messung die Farbwerte berechnet, so dass sie mit „getColor“ geholt werden können. Wenn keine Farbberechnung benötigt wird, kann durch die Deaktivierung ein wenig Zeit eingespart werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert, der bestimmt ob die Farbberechnung aktiviert wird: • true: Farbberechnung wird aktiviert • false: Farbberechnung wird deaktiviert

Rückgabe

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

5.7.3.2 GOMDBTS2048_isColorCalculation()

```
int __stdcall GOMDBTS2048_isColorCalculation (
             int handle,
             bool * value )
```

Ermittelt, ob die Farbberechnung aktiviert ist. Nur bei aktivierter „color calculation“, werden nach der spektralen Messung die Farbwerte berechnet, so dass sie mit „getColor“ geholt werden können.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert, enthält nach Rücksprung die Information, ob die Farbberechnung aktiviert ist, oder nicht • true: CT Limit Check aktiviert • false: CT Limit Check deaktiviert

Rückgabe

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

5.7.3.3 GOMDBTS2048_setColorCalculationMode()

```
int __stdcall GOMDBTS2048_setColorCalculationMode (
             int handle,
             int value )
```

Die Farbe kann auf Basis der einzelnen Pixel oder auf Basis des definierten Wellenlängenbereichs mit seinen Stützpunkten im Abstand der Schrittweite berechnet werden. Die Berechnung auf Basis der Pixel ist die genauest mögliche, kann aber geringfügig länger dauern als die Berechnung auf Basis der Wellenlängenstützpunkte.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert; enthält den Modus: • 0: Pixelbasiert • 1: Stützpunktbasiert

Rückgabe

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

5.7.3.4 GOMDBTS2048_getColorCalculationMode()

```
int __stdcall GOMDBTS2048_getColorCalculationMode (
            int handle,
            int * value )
```

Ermittelt den Modus, mit dem die Farbberechnungen durchgeführt werden. t

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer Wert; Farbberechnungsmodus: • 0: Pixelbasiert • 1: Stützpunktbasiert

Rückgabe

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

5.7.3.5 GOMDBTS2048_setColorCalculationOptimizationMode()

```
int __stdcall GOMDBTS2048_setColorCalculationOptimizationMode (
            int handle,
            int value )
```

Zur Berechnung von Farbdaten müssen einige Vorberechnungen durchgeführt werden. Diese Vorberechnungen sind sehr zeitintensiv. Daher können die Ergebnisse der Vorberechnungen können entweder für pixelbasierte oder stützpunktbasierte Berechnung im internen Speicher abgelegt werden. Dies spart zum tatsächlichen Zeitpunkt der Farbberechnung Zeit.
Wenn sowohl pixelbasierte Daten als auch stützpunktbasierte Daten ausgelesen werden sollen, dann muss in jedem Fall für eine der unterschiedlichen Datentypen nach der Messung diese Vorberechnung durchgeführt werden.
Wertebereich: 0 – keine Vorberechnung, 1 – pixelbasiert, 2 – stützpunktbasiert
Die stützpunktbasierte Vorberechnung wird, sofern sie aktiviert, ist bei jeder Änderung des Wellenlängenlängenbereiches neu durchgeführt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert; enthält den Modus: • 0: Inaktiv • 1: Pixelbasiert • 2: Stützpunktbasiert

Rückgabe

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

5.7.3.6 GOMDBTS2048_getColorCalculationOptimizationMode()

```
int __stdcall GOMDBTS2048_getColorCalculationOptimizationMode (
            int handle,
            int * value )
```

Ermittelt, ob die Berechnungsoptimierungen eingeschaltet wurden.

- 0: Optimierung nicht aktiv
- 1: Optimierung für pixelbasierte Berechnung aktiv
- 2: Optimierung für stützpunktbasierte Berechnung aktiv

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer Wert; Farbberechnungsoptimierung: • 0: Inaktiv • 1: Pixelbasiert • 2: Stützpunktbasiert

Rückgabe

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

5.7.3.7 GOMDBTS2048_setCTLimitCheckActive()

```
int __stdcall GOMDBTS2048_setCTLimitCheckActive (
            int handle,
            bool value )
```

Nur bei aktiviertem CT limit check wird die Farbtemperatur in Abhängigkeit des uv-Limits berechnet bzw. nicht berechnet. Mit dieser Methode kann der CT Limit Check aktiviert bzw. deaktiviert werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	active: Boolean Wert, der bestimmt ob der Limit Check aktiviert wird oder nicht • true: Limit Check wird aktiviert • false: Limit Check wird deaktiviert

Rückgabe

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

5.7.3.8 GOMDBTS2048_isCTLimitCheckActive()

```
int __stdcall GOMDBTS2048_isCTLimitCheckActive (
            int handle,
            bool * value )
```

Ermittelt, ob der CT Limit Check aktiviert ist. Abhängig von der Aktivierung wird die Farbtemperatur in Abhängigkeit vom zuvor gesetzten uv Limit berechnet oder nicht.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	active: Pointer auf Boolean Wert, enthält nach Rücksprung die Information, ob der CT Limit Check aktiviert ist: • true: Limit Check aktiviert • false: Limit Check deaktiviert

Rückgabe

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

5.7.3.9 GOMDBTS2048_setDeltaUVLimit()

```
int __stdcall GOMDBTS2048_setDeltaUVLimit (
            int handle,
            double limit )
```

Definiert die zulässigen uv-Limits innerhalb derer eine Berechnung der Farbtemperatur stattfinden soll. Wenn sich die tatsächlichen Werte außerhalb des Limits befinden, wird die Farbtemperatur nicht berechnet. Der Defaultwert ist „0.05“. Das Limit wird in Kombination mit der Methode „setCTLimitCheckActive“ aktiviert bzw. deaktiviert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*limit*	Double Wert, das Limit für die Berechnung der Farbtemperatur.

Rückgabe

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

5.7.3.10 GOMDBTS2048_getDeltaUVLimit()

```
int __stdcall GOMDBTS2048_getDeltaUVLimit (
              int handle,
              double * limit )
```

Liefert das zuvor gesetzte uv Limit, welches zur Berechnung der Farbtemperatur herangezogen wird. Das Limit ist nur aktiv wenn die Überprüfung mittel „setCTLimitCheckActive“ aktiviert bzw. deaktiviert wurde.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*limit*	Pointer auf Double Wert, enthält nach Rücksprung das definierte Limit.

Rückgabe

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

5.8 Messmethoden

Methoden um eine Messung vorzubereiten und durchzuführen.

Funktionen

- int __stdcall GOMDBTS2048_measure (int handle)
- int __stdcall GOMDBTS2048_spectralEvaluateIntegrationTimeInus (int handle, int *timeInus)
- int __stdcall GOMDBTS2048_measureGetCountsPixelFast (int handle, double *value)
- int __stdcall GOMDBTS2048_spectralMeasureOffset (int handle)
- int __stdcall GOMDBTS2048_spectralSaveStaticOffset (int handle)
- int __stdcall GOMDBTS2048_spectralLoadStaticOffset (int handle)
- int __stdcall GOMDBTS2048_spectralMeasureOffsetInDarkPosition (int handle)
- int __stdcall GOMDBTS2048_spectralMeasurePremeasuredOffset (int handle)
- int __stdcall GOMDBTS2048_spectralExportPremeasuredOffset (int handle, char *filename)
- int __stdcall GOMDBTS2048_spectralImportPremeasuredOffset (int handle, char *filename)
- int __stdcall GOMDBTS2048_spectralDeleteOffset (int handle)
- int __stdcall GOMDBTS2048_OORSLCorrectionMeasureFactors (int handle)
- int __stdcall GOMDBTS2048_integralMeasureOffset (int handle)
- int __stdcall GOMDBTS2048_integralMeasureOffsetInDarkPosition (int handle)
- int __stdcall GOMDBTS2048_setFilterPosition (int handle, int position)
- int __stdcall GOMDBTS2048_getFilterPosition (int handle, int *position)
- int __stdcall GOMDBTS2048_integralSeriesMeasure (int handle, int count)
- int __stdcall GOMDBTS2048_integralGetSeriesValues (int handle, double *values)
- int __stdcall GOMDBTS2048_setOut2LowDuringIntegralSeriesMeasurement (int handle, bool value)
- int __stdcall GOMDBTS2048_getOut2LowDuringIntegralSeriesMeasurement (int handle, bool *value)

5.8.1 Ausführliche Beschreibung

5.8.2 C++ Aufrufbeispiel

Spektralen Offset messen und Messung durchführen, siehe GOMDBTS2048_spectralMeasureOffset().

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                    //initialization
GOMDBTS2048_spectralSetOffsetMode(handle, 1);
GOMDBTS2048_spectralMeasureOffset(handle);
GOMDBTS2048_measure();                                     //start measure
    //do something
GOMDBTS2048_releaseHandle(handle);                       //release handle
```

5.8.3 Dokumentation der Funktionen

5.8.3.1 GOMDBTS2048_measure()

```
int __stdcall GOMDBTS2048_measure (
              int handle )
```

Diese Methode stößt die Messung an. Sie verwendet die zuvor gesetzten Einstellungen, wie z.B. Integrationszeit, Aktivierung der Messsensoren, Berechnung der Farbwerte . . . .
Nachdem die Messung durchgeführt wurde, können die gewünschten Messergebnisse mit den entsprechenden Methoden aus dem Gerät ausgelesen werden.
Wenn man sich im Messmodus „getriggerte Messung“ befindet, muss danach die Methode „isMeasurement↩ Finished“ solange aufgerufen werden, bis diese Methode „true“ zurückliefert. Erst zu diesem Zeitpunkt dürfen andere Methoden aufgerufen werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.2 GOMDBTS2048_spectralEvaluateIntegrationTimeInus()

```
int __stdcall GOMDBTS2048_spectralEvaluateIntegrationTimeInus (
            int handle,
            int * timeInus )
```

Diese Methode führt eine Testmessung durch und liefert die für das Spektrometer bei aktuell gesetztem Filter optimale Integrationszeit in der Einheit us zurück.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*timeInus*	Pointer auf Integer; enthält nach Rücksprung die Integrationszeit in [us].

Rückgabe

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

5.8.3.3 GOMDBTS2048_measureGetCountsPixelFast()

```
int __stdcall GOMDBTS2048_measureGetCountsPixelFast (
            int handle,
            double * value )
```

Diese Methode dient einer ultraschnellen Messung mit schnellem Datenausleseprozess ohne Berücksichtigung von Offsetwerten. Sie stößt eine spektrale Messung mit der eingestellten Integrationszeit an. Sämtliche weiteren zur Standardmessung verwendbaren Parameter werden bei dieser Methode nicht berücksichtigt. Nach der Messung werden die Counts der spektralen Einheit automatisch ausgelesen. Abhängig vom technischen Umfeld (LAN oder WLAN, PC, . . .) und der Integrationszeit können bei Verwendung dieser Methode für Messung und Datenausleseprozess Zeiten unter 5ms erreicht werden. Diese Methode ist bei Kommunikation über USB nicht verfügbar und gibt in diesem Falle einen Fehlercode zurück.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf das erste Element eines Double Array, enthält nach Rücksprung die Counts für jedes Pixel, das Array benötigt Platz für 2048 Double Werte.

Rückgabe

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

Aufrufbeispiel:

```
double* counts = new double[2048];
GOMDBTS2048_measureGetCountsPixelFast(handle, counts);
// do anything you like with the contents of your array e.g.:
cout << "pixel 0 = " << counts[0] << endl;
// ...
cout << "pixel 2047 = " << counts[2047] << endl;
delete [] counts;
```

5.8.3.4 GOMDBTS2048_spectralMeasureOffset()

```
int __stdcall GOMDBTS2048_spectralMeasureOffset (
              int handle )
```

Mit dieser Methode wird der spektrale Offset gemessen. Es wird die aktuell eingestellte Filterposition und Integrationszeit verwendet.

Achtung

Achtung: Wenn Sie den Dunkelstrom Offset messen wollen, müssen Sie zunächst das Filterrad auf „dunkel“ = Position 0 fahren. Besser jedoch ist, Sie verwenden die Methode spectralMeasureOffsetInDarkPosition().

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

Aufrufbeispiel:

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);
GOMDBTS2048_spectralSetOffsetMode(handle, 1);
GOMDBTS2048_spectralMeasureOffset(handle);
GOMDBTS2048_measure();
GOMDBTS2048_releaseHandle(handle);
```

5.8.3.5 GOMDBTS2048_spectralSaveStaticOffset()

```
int __stdcall GOMDBTS2048_spectralSaveStaticOffset (
              int handle )
```

Mit dieser Methode können Sie den statischen Offset für eine spätere Messung hinterlegen. Es werden die Dunkelcounts und die dazugehörige Integrationszeit abgespeichert. Es können beliebig viele Offsetmessungen hinterlegt werden. Mit releaseHandle() werden alle Offset Messungen wieder gelöscht und der Speicher freigegeben.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.6 GOMDBTS2048_spectralLoadStaticOffset()

```
int __stdcall GOMDBTS2048_spectralLoadStaticOffset (
              int handle )
```

Diese Methode lädt einen zuvor gespeicherten statischen Offset. Falls für die aktuelle Integrationszeit noch kein Offset hinterlegt wurde gibt diese Methode den Fehlercode -15076 zurück.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.7 GOMDBTS2048_spectralMeasureOffsetInDarkPosition()

```
int __stdcall GOMDBTS2048_spectralMeasureOffsetInDarkPosition (
              int handle )
```

Diese Methode funktioniert prinzipiell wie die Methode spectralMeasureOffset(). Allerdings wird der Offset in der „dunkel“-Position des Filterrads gemessen. Danach fährt der Filter in die ursprüngliche Position zurück.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.8 GOMDBTS2048_spectralMeasurePremeasuredOffset()

```
int __stdcall GOMDBTS2048_spectralMeasurePremeasuredOffset (
              int handle )
```

Mit dieser Methode wird der "Premeasured Offset" für fest definierte Integrationszeiten gemessen. Dieser wird für den für den „vorgemessenen Offset Modus“ benötigt. Dabei wird immer die Dunkelposition des Filters verwendet. Danach fährt der Filter in die ursprüngliche Position zurück. Der Aufruf dieser Methode kann je nach Gerät zwischen 15 Sekunden (BTS2048-VL) und 120 Sekunden (z.B. BTS2048-UV) Sekunden dauern.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.9 GOMDBTS2048_spectralExportPremeasuredOffset()

```
int __stdcall GOMDBTS2048_spectralExportPremeasuredOffset (
            int handle,
            char * filename )
```

Mit dieser Methode kann der "Premeasured Offset" für einen späteren Zeitpunkt in einer Datei gespeichert werden. Es wird aber empfohlen den Offset regelmäßig neu zu messen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*filename*	Nullterminierter String, enthält den kompletten Pfad auf die zu ladende Datei.

Rückgabe

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

5.8.3.10 GOMDBTS2048_spectralImportPremeasuredOffset()

```
int __stdcall GOMDBTS2048_spectralImportPremeasuredOffset (
            int handle,
            char * filename )
```

Mit dieser Methode kann der "Premeasured Offset" aus einer Datei auf der Festplatte geladen werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*filename*	Nullterminierter String, enthält den kompletten Pfad auf die zu ladende Datei.

Rückgabe

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

5.8.3.11 GOMDBTS2048_spectralDeleteOffset()

```
int __stdcall GOMDBTS2048_spectralDeleteOffset (
            int handle )
```

Mit dieser Methode wird der aktuell im Gerät befindliche Offset zurückgesetzt.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

Aufrufbeispiel:

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);
GOMDBTS2048_spectralSetOffsetMode(handle, 1);
GOMDBTS2048_spectralMeasureOffset(handle);
GOMDBTS2048_measure();
GOMDBTS2048_spectralDeleteOffset(handle);
GOMDBTS2048_releaseHandle(handle);
```

5.8.3.12 GOMDBTS2048_OORSLCorrectionMeasureFactors()

```
int __stdcall GOMDBTS2048_OORSLCorrectionMeasureFactors (
            int handle )
```

Diese Methode steht nur bei OOR SLC (Out of Range Straylight Correction) Kalibriereinträgen zur Verfügung (ausschließlich bei BTS2048-UV Geräten).
Sie misst die OOR SL-Correction Faktoren welche mit der Methode OORSLCorrectionSetMode() genutzt werden können.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.13 GOMDBTS2048_integralMeasureOffset()

```
int __stdcall GOMDBTS2048_integralMeasureOffset (
            int handle )
```

Mit dieser Methode wird der integrale Offset gemessen. Es wird die aktuell eingestellte Filterposition und Integrationszeit verwendet.
Achtung:
Wenn Sie den Dunkelstrom Offset messen wollen, müssen Sie zunächst das Filterrad auf „dunkel“ = Position 0 fahren. Besser jedoch ist, Sie verwenden die Methode integralMeasureOffsetInDarkPosition().

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.14 GOMDBTS2048_integralMeasureOffsetInDarkPosition()

```
int __stdcall GOMDBTS2048_integralMeasureOffsetInDarkPosition (
            int handle )
```

Diese Methode misst den integralen Offset in der „dunkel“-Position des Filterrads. Danach fährt der Filter in die ursprüngliche Position zurück. Der integrale Offset wird bei der Geräte initialisierung automatisch gemessen. Da er unabhängig von der integralem Messzeit ist, muss er nur dann manuell gemessen werden, wenn ein sehr schwaches Signal gemessen wird.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.8.3.15 GOMDBTS2048_setFilterPosition()

```
int __stdcall GOMDBTS2048_setFilterPosition (
            int handle,
            int position )
```

Legt die Position des Filters manuell fest.
BTS2048-VL: Ihr BTS2048 enthält ein Filterrad mit bis zu 4 unterschiedlichen Filtern zum Dämpfen des Signals.
Wertebereich: 0 – 3, wobei gilt:

- 0: Sperrfilter,
- 1: OD2,
- 2: OD1,
- 3: kein Filter.

BTS2048-UV: Ihr Gerät enthält bis zu 8 unterschiedliche Filter.
Wertebereich: 0 – 7, wobei gilt:

- 0: Sperrfilter,
- 1: kein Filter,
- 2: OoR Filter
- 3: BP Filter 1
- 4: BP Filter 2
- 5: BP Filter 3
- 6: BP Filter 4
- 7: BP Filter 5

Diese Filterposition wird für die Messung verwendet und überschreibt die durch den Kalibriereintrag definierte Filterposition.
Wenn der gesetzte Filter nicht zum ausgewählten Kalibriereintrag passt, können lediglich die Counts ausgelesen werden, da die kalibrierten Werte dann ungültig sind.
Wenn ein neuer Kalibriereintrag ausgewählt wird, wird wieder die Filterposition zur Messung verwendet, die im Kalibriereintrag hinterlegt ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*position*	Integerwert; enthält die Filternummer, die verwendet werden soll.

Rückgabe

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

5.8.3.16 GOMDBTS2048_getFilterPosition()

```
int __stdcall GOMDBTS2048_getFilterPosition (
            int handle,
            int * position )
```

Ihr BTS2048 enthält ein Filterrad mit bis zu 4 (BTS2048-VL) bzw. 8 (BTS2048-UV) unterschiedlichen Filtern zum Dämpfen des Signals. Wertebereich: 0-3 bzw. 0-8, wobei Position 0 im Normalfall die Position ohne Filter ist. Diese Methode ermittelt die aktuell gesetzte Filterradposition.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*position*	Pointer auf Integerwert; enthält die aktuell gesetzte Filterpositionsnummer

Rückgabe

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

5.8.3.17 GOMDBTS2048_integralSeriesMeasure()

```
int __stdcall GOMDBTS2048_integralSeriesMeasure (
            int handle,
            int count )
```

Mit dieser Methode wird eine Serie an integralen Messungen durchgeführt. Die Anzahl der Messungen wird mit dem Parameter count festgelegt und jede Messung wird mit einem einzelnen Triggerpuls gestartet. Der Trigger Input muss auf 1 oder 3 gesetzt und der integrale Messbereich muss fest eingestellt sein. (kein Autorange) Bevor die Messung gestartet wird müssen die Triggereinstellungen (high/low, Pegel/Flanke) mit den Methoden setTrigger↩ Mode() und setTriggerLevel() eingestellt werden. Nachdem alle Messungen abgeschlossen sind, kommt diese Methode zurück und die Messwerte können mit der Methode integralGetSeriesValues() ausgelesen werden. Beachten Sie, dass die Dauer zwischen den Triggerpulsen größer als die eingestellte integrale Integrationszeit sein muss.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*count*	Integerwert; Anzahl Messungen die durchgeführt werden sollen.

Rückgabe

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

5.8.3.18 GOMDBTS2048_integralGetSeriesValues()

```
int __stdcall GOMDBTS2048_integralGetSeriesValues (
            int handle,
            double * values )
```

Diese Methode ließt die Messwerte aus, die mit der Methode integralSeriesMeasure() aufgenommen wurden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*values*	Pointer auf das erste Element eines Double Array, enthält die Messwerte, die Größe des Array muss mindestens die Größe haben welche mit integralSeriesMeasure() vordefiniert wurde.

Rückgabe

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

5.8.3.19 GOMDBTS2048_setOut2LowDuringIntegralSeriesMeasurement()

```
int __stdcall GOMDBTS2048_setOut2LowDuringIntegralSeriesMeasurement (
            int handle,
            bool value )
```

Diese Methode setzt den Ausgang 2 des Geräts auf Low während eine Messung der Integralen Serie durchgeführt wird.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	boolean Wert: false -> aus, true -> Ausgang 2 low während der Messung

Rückgabe

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

5.8.3.20 GOMDBTS2048_getOut2LowDuringIntegralSeriesMeasurement()

```
int __stdcall GOMDBTS2048_getOut2LowDuringIntegralSeriesMeasurement (
            int handle,
            bool * value )
```

Diese Methode prüft, ob der Ausgang 2 während der Integralen Serienmessung auf low gesetzt ist.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Pointer auf boolean Wert: false -> aus, true -> output 2 low während der Messung

Rückgabe

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

5.9 Asynchrone Messmethoden

Methoden um eine asynchrone Messung durchzuführen.

Funktionen

- int __stdcall GOMDBTS2048_asyncStartMeasurement (int handle)
- int __stdcall GOMDBTS2048_asyncStartMeasurementWithTime (int handle, double *time)
- int __stdcall GOMDBTS2048_asyncGetProgress (int handle, bool *finished, int *progress)
- int __stdcall GOMDBTS2048_asyncStopMeasurement (int handle)

5.9.1 Ausführliche Beschreibung

5.9.2 C++ Aufrufbeispiel

Statischen Offset messen und asychrone Messung starten.

```
int handle, progress;
bool finished = false;
GOMDBTS2048_getHandle(NULL, &handle);                   //initialization
GOMDBTS2048_spectralSetOffsetMode(handle, 1);
GOMDBTS2048_spectralMeasureOffset(handle);
GOMDBTS2048_asyncStartMeasurement();                    //start measure
while(!finished)
{
    GOMDBTS2048_asyncGetProgress(handle, &finished, &progress);
}
GOMDBTS2048_releaseHandle(handle);                      //release handle
```

5.9.3 Dokumentation der Funktionen

5.9.3.1 GOMDBTS2048_asyncStartMeasurement()

```
int __stdcall GOMDBTS2048_asyncStartMeasurement (
            int handle )
```

Diese Methode stößt eine asynchrone Messung an. Sie funktioniert genau so wie die standard Messmethode "measure()" kehrt sofort zurück und startet die Messung in einem seperaten Hintergrund-Prozess. Wenn SIe diese Methode verwenden, wird ein Grundwissen über Multithreading vorausgesetzt. Der Fortschritt der Messung can mit der Methode asyncGetProgress() abgerufen werden. Es ist vorgesehen, diese Mehtode solange hintereinander aufzurufen, bis der Parameter finished den Wert "true" zurück liefert. Der Hintergrundprozess wird dann automatisch mit dem Haupprozess zusammengeführt.
Die asynchrone Messung is mit der getriggerten Messung nicht kompatibel. Nachdem die Messung durchgeführt wurde, können die gewünschten Messergebnisse mit den entsprechenden Methoden aus dem Gerät ausgelesen werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.9.3.2 GOMDBTS2048_asyncStartMeasurementWithTime()

```
int __stdcall GOMDBTS2048_asyncStartMeasurementWithTime (
             int handle,
             double * time )
```

Diese Methode stößt ebenfalls eine asynchrone Messung an, sie liefert aber zusätzlich die erwartete Messdauer zurück. Aus diesem Grund kommt die Methode nicht unbedingt direkt nach Messaufruf zrück. Zunächst wird die Messzwit berechnet und wenn der dynamicTimeMode aktiviert ist, wird die Vormessung zur Bestimmung der Integrationszeit for dem Methodenrücksprung durchgeführt.
Behalten Sie im Kopf, dass in manchen Situationen eine Bestimmung der erwarteten Messdauer im Vorfeld nicht möglich ist. (z.B. bei Multi-Messungen mit dem BTS2048-UV) In diesem Fall kann die Zeit lediglich als Anhaltspunkt verwendet werden.
Die Verwendung der Methode ist ansonsten gleich der wie bei asyncStartMeasurement().

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*time*	Pointer auf Doublewert; enthält die voraussichtiche Messdauer in Sekunden

Rückgabe

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

5.9.3.3 GOMDBTS2048_asyncGetProgress()

```
int __stdcall GOMDBTS2048_asyncGetProgress (
             int handle,
             bool * finished,
             int * progress )
```

Während einer asynchronen Messung kann mit dieser Methode der Status der Messung abgefragt werden. Außerdem führt sie den Hintergrunfprozess der Messung mit demHauptprozess zusammen. Aus diesem Grund ist es auch unbedingt nötig diese Funktion solange aufzurufen, bis der Parameter finsihed mindestens ein mal den Wert "true" zurückliefert.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*finished*	Pointer auf Booleanwert; Wurde die Messung abgeschlossen?
out	*progress*	Pointer auf Integerwert; Fortschitt der Messung in %. Wird nur in unregelmäßigen Abständen aktualisiert.

Rückgabe

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

5.9.3.4 GOMDBTS2048_asyncStopMeasurement()

```
int __stdcall GOMDBTS2048_asyncStopMeasurement (
              int handle )
```

Mit dieser Methode kann eine asynchrone Messung vor der eigentlichen Beendigung abgebrochen werden. Allerdings hat die Methode keinen Einfluss auf den Messvorgang im Gerät, deshalb kann es einige Zeit dauern bis die Messung tatsächlich unterbrochen wurde.
Auch bei Verwendung dieser Methode ist ein Anschließender Aufruf der Methode asyncGetProgress() bis der Parameter finished den Wert "true" liefert unbedingt notwendig. Außderdem kann so der Stand des Messabbruchs abgerufen werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.10 Methoden zum auslesen der integralen Messwerte

Methods um die integralen Messergebnisse einer zuvor durchgeführten Messung auszulesen.

Funktionen

- int __stdcall GOMDBTS2048_integralGetUnit (int handle, int calibrationEntryNumber, char ∗unit)
- int __stdcall GOMDBTS2048_integralGetSaturation (int handle, double ∗saturationPercent, double ∗saturation15bit)
- int __stdcall GOMDBTS2048_integralGetLastUsedAz (int handle, double ∗az)
- int __stdcall GOMDBTS2048_integralGetValue (int handle, double ∗value)
- int __stdcall GOMDBTS2048_integralGetCurrent (int handle, double ∗value)
- int __stdcall GOMDBTS2048_integralGetLastUsedRange (int handle, int ∗range)

5.10.1 Ausführliche Beschreibung

5.10.2 C++ Aufrufbeispiel

Sättigung vom integratlen Sensors ermitteln, siehe GOMDBTS2048_integralGetSaturation().

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                  //initialization
GOMDBTS2048_spectralSetOffsetMode(handle, 1);
GOMDBTS2048_spectralMeasureOffset(handle);
GOMDBTS2048_measure();                                   //start measure
//do something
GOMDBTS2048_releaseHandle(handle);                     //release handle
```

5.10.3 Dokumentation der Funktionen

5.10.3.1 GOMDBTS2048_integralGetUnit()

```
int __stdcall GOMDBTS2048_integralGetUnit (
            int handle,
            int calibrationEntryNumber,
            char * unit )
```

Ermittelt die SI-Einheit, die zu einer Kalibrierung des integralen Sensors gehört. Es gibt maximal 52 Kalibriereinträge, die aber nicht alle belegt sein müssen. Kalibriereinträge, die nicht belegt sind, liefern einen Leerstring. Es muss genügend Speicher (max. Länge: 20 Bytes) für den Antwortstring bereitgestellt werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, die Nummer des Kalibriereintrags für den die Einheit ermittelt werden soll; valid range of values: 0 - 51
out	*unit*	Nullterminierter String; beinhaltet nach Rücksprung die SI-Einheit.

Rückgabe

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

5.10.3.2 GOMDBTS2048_integralGetSaturation()

```
int __stdcall GOMDBTS2048_integralGetSaturation (
            int handle,
            double * saturationPercent,
            double * saturation15bit )
```

Ermittelt die Sättigung des integralen Sensors im eingestellten Rangebereich der letzten durchgeführten Messung. Die Rückgabe des Ergebnisses erfolgt zum einen in Prozent zum anderen auf Basis einer 15bit Zahl. 0% entsprechen dabei dem Wert 0. 100% entsprechen dem Wert 32768.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*saturationPercent*	Pointer auf einen Double Wert, enthält nach Rücksprung die Sättigung in Prozent;
out	*saturation15bit*	Pointer auf einen Double Wert, enthält nach Rücksprung die Sättigung; Bezugsgröße: 100% entsprechen 32768

Rückgabe

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

5.10.3.3 GOMDBTS2048_integralGetLastUsedAz()

```
int __stdcall GOMDBTS2048_integralGetLastUsedAz (
            int handle,
            double * az )
```

Ermittelt den zuletzt verwendeten a(z) Korrekturfaktor. Dieser Faktor wird auf Basis des letzten gemessenen Spektrums berechnet, statisch definiert oder entspricht dem Wert 1.0, falls der a(z)-Modus auf den Wert „0“ gesetzt wird. Der Korrekturfaktor dient der spektralen Anpassung des integralen Sensors.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*az*	Pointer auf einen Double Wert, enthält nach Rücksprung den a(z) Korrekturfaktor

Rückgabe

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

5.10.3.4 GOMDBTS2048_integralGetValue()

```
int __stdcall GOMDBTS2048_integralGetValue (
              int handle,
              double * value )
```

Diese Methode liefert den mit dem Kalibrierfaktor und a(z)-Korrekturfaktor beaufschlagten integralen Messwert. Je nach Systemaufbau und entsprechend selektiertem Kalibriertabelleneintrag handelt es sich hier um die Beleuchtungsstärke, Lichtstärke oder Lichtstrom.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Double Wert, Messwert des integralen Sensors.

Rückgabe

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

5.10.3.5 GOMDBTS2048_integralGetCurrent()

```
int __stdcall GOMDBTS2048_integralGetCurrent (
              int handle,
              double * value )
```

Diese Methode liefert den Rohwert (Stromstärke in [A]) der integralen Messung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Double Wert, gemessene Stromstärke des integralen Sensors

Rückgabe

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

5.10.3.6 GOMDBTS2048_integralGetLastUsedRange()

```
int __stdcall GOMDBTS2048_integralGetLastUsedRange (
              int handle,
              int * range )
```

Diese Methode liefert den integralen Messbereich der für die letzte Messung verwendet wurde. Dies ist hilfreich, wenn autorange aktiviert ist.

Parameter

`in`	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
`out`	*range*	Pointer auf Double Wert, letzter verwendeter integraler Messbereich

Rückgabe

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

5.11 Methoden zum auslesen der spektralen Messwerte

Methods um die spektralen Messergebnisse einer zuvor durchgeführten Messung auszulesen.

Funktionen

- int __stdcall GOMDBTS2048_spectralGetUnit (int handle, int calibrationEntryNumber, char ∗unit)
- int __stdcall GOMDBTS2048_spectralGetSaturation (int handle, double ∗saturation)
- int __stdcall GOMDBTS2048_getRadiometricValueOverWLRange (int handle, double ∗value)
- int __stdcall GOMDBTS2048_getPeak (int handle, double ∗lambda, double ∗power)
- int __stdcall GOMDBTS2048_getFWHM (int handle, double ∗fwhm)
- int __stdcall GOMDBTS2048_getCenterWavelength (int handle, double ∗value)
- int __stdcall GOMDBTS2048_getCentroidWavelength (int handle, double ∗value)
- int __stdcall GOMDBTS2048_spectralGetSpectrumCalibratedWavelength (int handle, double ∗spectrum)
- int __stdcall GOMDBTS2048_spectralGetSpectrumCalibratedPixel (int handle, double ∗spectrum)
- int __stdcall GOMDBTS2048_spectralGetCountsPixel (int handle, double ∗counts)
- int __stdcall GOMDBTS2048_spectralGetLambdas (int handle, bool wavelengthRaster, double ∗lambdas)
- int __stdcall GOMDBTS2048_spectralGetSpecmax (int handle, int ∗value)
- int __stdcall GOMDBTS2048_spectralGetLastUsedOffset (int handle, double ∗values)

5.11.1 Ausführliche Beschreibung

5.11.2 C++ Aufrufbeispiel

Messresultate des Spektrometers ermitteln und ausgeben, siehe GOMDBTS2048_spectralGetSpectrum↩ CalibratedPixel().

```
int handle;
double values [2048];
GOMDBTS2048_getHandle(NULL, &handle);                                  //initialization
GOMDBTS2048_measure();                                                   //start measure
GOMDBTS2048_spectralGetSpectrumCalibratedPixel(handle, values
      );       //returns calibrated pixel
// do anything you like with the contents of your array e.g.:
cout <<"first element = " << values[0] << endl;                         //writes on console
//...
cout <<"last element = "<< values[2047] << endl;

GOMDBTS2048_releaseHandle(handle);                                     //release
       handle
```

5.11.3 Dokumentation der Funktionen

5.11.3.1 GOMDBTS2048_spectralGetUnit()

```
int __stdcall GOMDBTS2048_spectralGetUnit (
             int handle,
             int calibrationEntryNumber,
             char * unit )
```

Ermittelt die SI-Einheit, die zu einer Kalibrierung des spektralen Sensors gehört. Es gibt maximal 52 Kalibriereinträge, die aber nicht alle belegt sein müssen. Kalibriereinträge, die nicht belegt sind, liefern einen Leerstring. Es muss genügend Speicher (max. Länge: 20 Bytes) für den Antwortstring bereitgestellt werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*calibrationEntryNumber*	Integer Wert, die Nummer des Kalibriereintrags für den die Einheit ermittelt werden soll; gültiger Wertebereich: 0 – 51.
out	*unit*	nullterminierter String; beinhaltet nach Rücksprung die SI-Einheit.

Rückgabe

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

5.11.3.2 GOMDBTS2048_spectralGetSaturation()

```
int __stdcall GOMDBTS2048_spectralGetSaturation (
            int handle,
            double * saturation )
```

Liefert die spektrale Aussteuerung der letzten durchgeführten Messung

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*saturation*	Pointer auf Integer Wert, enthält nach Rücksprung die spektrale Aussteuerung der letzten Messung in Prozent.

Rückgabe

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

5.11.3.3 GOMDBTS2048_getRadiometricValueOverWLRange()

```
int __stdcall GOMDBTS2048_getRadiometricValueOverWLRange (
            int handle,
            double * value )
```

Diese Methode liefert den radiometrischen Wert einer vorhergehenden Messung, durch die Berechnung eines Integrals über den definierten Wellenlängenbereich (setWavelengthRange()).

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Double Wert, enthält “groß” X.

Rückgabe

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

5.11.3.4 GOMDBTS2048_getPeak()

```
int __stdcall GOMDBTS2048_getPeak (
             int handle,
             double * lambda,
             double * power )
```

Liefert den Spitzenwert der spektralen Messung. Es wird sowohl der Messwert als auch die zugehörige Wellenlänge in [nm] ermittelt. Die spektrale Einheit hängt vom selektierten Kalibriertabelleneintrag ab und kann mit der Methode “spectralGetUnit” ermittelt werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*lambda*	Pointer auf Double Wert, enthält die Wellenlänge in [nm]
out	*power*	Pointer auf Double, enthält den Spitzenwert der Messung

Rückgabe

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

5.11.3.5 GOMDBTS2048_getFWHM()

```
int __stdcall GOMDBTS2048_getFWHM (
             int handle,
             double * fwhm )
```

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

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*fwhm*	Pointer auf Double Wert, enthält die Halbwertsbreite in [nm]

Rückgabe

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

5.11.3.6 GOMDBTS2048_getCenterWavelength()

```
int __stdcall GOMDBTS2048_getCenterWavelength (
             int handle,
             double * value )
```

Liefert den Mittelpunkt der FWHM in [nm] nach einer ausgeführten Messung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Double Wert, enthält den Mittelpunkt der FWHM in [nm]

Rückgabe

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

5.11.3.7 GOMDBTS2048_getCentroidWavelength()

```
int __stdcall GOMDBTS2048_getCentroidWavelength (
             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 BTS2048; 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.

5.11.3.8 GOMDBTS2048_spectralGetSpectrumCalibratedWavelength()

```
int __stdcall GOMDBTS2048_spectralGetSpectrumCalibratedWavelength (
             int handle,
             double * spectrum )
```

Ermittelt die mit den Kalibrierfaktoren und Substitutionsfaktoren beaufschlagten Messresultate des Spektrometers. Die Anzahl der zu erwartenden Werte sollte vorher mit der Methode “spectralGetSpecmax” ermittelt werden, um genügend Speicher für das Ergebnisarray zur Verfügung zu stellen.
Das Erste Ergebniselement enthält den Messwert für die definierte Startwellenlänge.
Das Zweite Ergebniselement enthält den Messwert für die definierte Startwellenlänge + definierte Schrittweite.
....
Die Interpolationspunkte werden durch die Methode “setWavelengthRange” definiert, wobei die Startwellenlänge, die Endwellenlänge und die Schrittweite definiert werden können.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; 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 Array muss vor Aufruf definiert werden und hängt von dem zuvor definierten Wellenlängenbereich und der Schrittweite ab.

Rückgabe

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

5.11.3.9 GOMDBTS2048_spectralGetSpectrumCalibratedPixel()

```
int __stdcall GOMDBTS2048_spectralGetSpectrumCalibratedPixel (
            int handle,
            double * spectrum )
```

Ermittelt die mit den Kalibrierfaktoren und Substitutionsfaktoren beaufschlagten Messresultate des Spektrometers. Es werden 2048 Werte zurückgegeben entsprechend den 2048 Pixeln des Arrays.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; 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 Array ist 2048

Rückgabe

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

5.11.3.10 GOMDBTS2048_spectralGetCountsPixel()

```
int __stdcall GOMDBTS2048_spectralGetCountsPixel (
            int handle,
            double * counts )
```

Liefert die Rohwerte des gemessenen Spektrums. Basis sind die einzelnen “Counts” für jedes Pixel. Es muss ein Double Array der Größe 2048 bereitgestellt werden, in welchem die Ergebnisse zurückgegeben werden.

- Das erste Ergebniselement enthält die “Counts” des ersten Pixels.
- Das zweite Ergebniselement enthält die “Counts” des zweiten Pixels.
- ...

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*counts*	Pointer auf das erste Element eines Double Array, enthält nach Rücksprung die Counts für jedes Pixel, das Array benötigt Platz für 2048 Double Werte.

Rückgabe

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

5.11.3.11 GOMDBTS2048_spectralGetLambdas()

```
int __stdcall GOMDBTS2048_spectralGetLambdas (
            int handle,
            bool wavelengthRaster,
            double * lambdas )
```

Liefert die Wellenlängen zurück.
Wenn „wavelengthRaster“ auf true gesetzt wird, erhält man die Wellenlängen bzgl. des eingestellten „Wavelength↩ Range“ zurück. Die Größe des Double Array ist abhängig von der Startwellenlänge, Endwellenlänge und der Schrittweite und kann mit „spectralGetSpecmax“ ermittelt werden.
Ist „wavlengthRaster“ = false, dann erhält man die Wellenlängenzuordnung zu den 2048 Pixeln der spektralen Messeinheit. In diesem Fall muss ein Double Array der Größe 2048 bereitgestellt werden, in welchem die Ergebnisse zurückgegeben werden.

- Das erste Ergebniselement enthält die “Wellenlänge” des ersten Pixels.
- Das zweite Ergebniselement enthält die “Wellenlänge” des zweiten Pixels.
- ...

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*wavelengthRaster*	Boolean Wert; • true: Wellenlängen bzgl. des eingestellten „WavelengthRange“ • false: Wellenlängenzuordnung zu den 2048 Pixeln der spektralen Messeinheit
out	*lambdas*	Pointer auf das erste Element eines Double Array, enthält nach Rücksprung die Counts für jedes Pixel, das Array benötigt Platz für 2048 Double Werte.

Rückgabe

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

5.11.3.12 GOMDBTS2048_spectralGetSpecmax()

```
int __stdcall GOMDBTS2048_spectralGetSpecmax (
            int handle,
            int * value )
```

Diese Methode liefert Ihnen die Anzahl der zu erwartenden Elemente beim Aufruf der Methoden “spectralGet↩ CountsWavelength” oder “spectralGetSpectrumCalibratedWavelength” zurück. Die Anzahl variiert je nach definiertem Wellenlängenbereich und der eingestellten Schrittweite.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integerwert, enthält die Anzahl der zu erwartenden Elemente.

Rückgabe

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

5.11.3.13 GOMDBTS2048_spectralGetLastUsedOffset()

```
int __stdcall GOMDBTS2048_spectralGetLastUsedOffset (
            int handle,
            double * values )
```

Liefert die Dunkelcount der letzten durchgeführten spektralen Messung, falls diese vorhanden sind.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*values*	Pointer auf das erste Element eines Double Array, enthält nach Rücksprung die Dunkelcounts, das Array benötigt Platz für 2048 Double Werte.

Rückgabe

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

5.12 Methoden zum auslesen von allgemeinen Messwerten

Funktionen

- int __stdcall GOMDBTS2048_getTemperature (int handle, double ∗value)
- int __stdcall GOMDBTS2048_getLastMaxADC (int handle, int ∗value)
- int __stdcall GOMDBTS2048_getLastScaleWithVLFactor (int handle, double ∗value)

5.12.1 Ausführliche Beschreibung

5.12.2 Dokumentation der Funktionen

5.12.2.1 GOMDBTS2048_getTemperature()

```
int __stdcall GOMDBTS2048_getTemperature (
            int handle,
            double * value )
```

Misst die aktuelle Temperatur des Temperatursensors auf der Elektronik. Die Einheit beträgt [℃].

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Double Wert; enthält die Temperatur in [℃].

Rückgabe

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

5.12.2.2 GOMDBTS2048_getLastMaxADC()

```
int __stdcall GOMDBTS2048_getLastMaxADC (
            int handle,
            int * value )
```

Liefert den maximalen Count der letzten durchgeführten spektralen Messung. Vom Rückgabewert ist der Offset bereits abgezogen.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer, enthält nach Rücksprung den maximalen counts der letzten durchgeführten spektralen Messung.

Rückgabe

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

5.12.2.3 GOMDBTS2048_getLastScaleWithVLFactor()

```
int __stdcall GOMDBTS2048_getLastScaleWithVLFactor (
            int handle,
            double * value )
```

Liefert den Skalierungsfaktor der letzten spektralen Messung. Für mehr Information siehe die Methode setScale↩ WithVL()

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Double, enthält nach Rücksprung den letzten "skaliere mit VL" Faktor

Rückgabe

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

5.13 Methoden zum auslesen der Farbwerte

Funktionen

- int __stdcall GOMDBTS2048_calculateColor (int handle)
- int __stdcall GOMDBTS2048_getColor (int handle, double *UpperX, double *UpperY, double *UpperZ, double *x, double *y, double *us, double *vs, double *CCT, double *domWL)
- int __stdcall GOMDBTS2048_getDeltaUV (int handle, double *uv)
- int __stdcall GOMDBTS2048_getPurity (int handle, double *value)
- int __stdcall GOMDBTS2048_getCRI (int handle, double *Ra, double *R1, double *R2, double *R3, double *R4, double *R5, double *R6, double *R7, double *R8, double *R9, double *R10, double *R11, double *R12, double *R13, double *R14, double *R15)

5.13.1 Ausführliche Beschreibung

5.13.2 C++ Aufrufbeispiel

Messen und auslesen von Farbwerten.

```
int handle;
double X, Y, Z, x, y, us, vs, cct, domWL;
GOMDBTS2048_getHandle(NULL, &handle);                          //initialization
GOMDBTS2048_setColorCalculation(handle, true);
GOMDBTS2048_measure();                                          //start measure
GOMDBTS2048_getColor(handle, &X, &Y, &Z, &x, &y, &us, &vs, &cct, &domWL);
GOMDBTS2048_releaseHandle(handle);                             //release handle
```

5.13.3 Dokumentation der Funktionen

5.13.3.1 GOMDBTS2048_calculateColor()

```
int __stdcall GOMDBTS2048_calculateColor (
            int handle )
```

Wenn die Berechnung der Farbwerte nicht aktiv sein sollte (setColorCalculation(false)), kann man die Berechnung manuell anstoßen. Dies ist nicht notwendig wenn die Berechnung aktiv ist (setColorCalculation(true)). In diesem Falle wird die Berechnung direkt nach der Messung automatisch durchgeführt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.13.3.2 GOMDBTS2048_getColor()

```
int __stdcall GOMDBTS2048_getColor (
            int handle,
```

```
            double * UpperX,
            double * UpperY,
            double * UpperZ,
            double * x,
            double * y,
            double * us,
            double * vs,
            double * CCT,
            double * domWL )
```

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

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*UpperX*	Pointer auf Double Wert, enthält “groß” X
out	*UpperY*	Pointer auf Double Wert, enthält “groß” Y
out	*UpperZ*	Pointer auf Double Wert, enthält “groß” Z
out	*x*	Pointer auf Double Wert, enthält x entsprechend dem CIE1931 Farbraum
out	*y*	Pointer auf Double Wert, enthält y entsprechend dem CIE1931 Farbraum
out	*us*	Pointer auf Double Wert, enthält u‘ entsprechend dem CIE1976 Farbraum
out	*vs*	Pointer auf Double Wert, enthält v‘ entsprechend dem CIE1976 Farbraum
out	*CCT*	Pointer auf Double Wert, „Correlated Color Temperature“, die Farbtermperatur in [K]
out	*domWL*	Pointer auf Double Wert, enthält die dominante Wellenlänge

Rückgabe

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

Aufrufbeispiel:

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);
GOMDBTS2048_setColorCalculation(handle, true);
GOMDBTS2048_measure();
double X, Y, Z, x, y, us, vs, cct, domWL;
GOMDBTS2048_getColor(handle, &X, &Y, &Z, &x, &y, &us, &vs, &cct, &domWL);
GOMDBTS2048_releaseHandle(handle);
```

5.13.3.3 GOMDBTS2048_getDeltaUV()

```
int __stdcall GOMDBTS2048_getDeltaUV (
            int handle,
            double * uv )
```

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

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*uv*	Pointer auf Double Wert, enthält nach Rücksprung den ermittelten delta uv Wert.

Rückgabe

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

5.13.3.4 GOMDBTS2048_getPurity()

```
int __stdcall GOMDBTS2048_getPurity (
             int handle,
             double * value )
```

Liefert die Farbreinheit der letzten Messung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Double Wert, enthält nach Rücksprung die ermittelte Farbreinheit.

Rückgabe

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

5.13.3.5 GOMDBTS2048_getCRI()

```
int __stdcall GOMDBTS2048_getCRI (
             int handle,
             double * Ra,
             double * R1,
             double * R2,
             double * R3,
             double * R4,
             double * R5,
             double * R6,
             double * R7,
             double * R8,
             double * R9,
             double * R10,
             double * R11,
             double * R12,
             double * R13,
             double * R14,
             double * R15 )
```

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 BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*Ra*	Pointer auf Double Wert, Durchschnittswert von R1 – R8
out	*R1*	Pointer auf Double Wert, Wert von R1
out	*R2*	Pointer auf Double Wert, Wert von R2
out	*R3*	Pointer auf Double Wert, Wert von R3
out	*R4*	Pointer auf Double Wert, Wert von R4
out	*R5*	Pointer auf Double Wert, Wert von R5
out	*R6*	Pointer auf Double Wert, Wert von R6
out	*R7*	Pointer auf Double Wert, Wert von R7
out	*R8*	Pointer auf Double Wert, Wert von R8
out	*R9*	Pointer auf Double Wert, Wert von R9
out	*R10*	Pointer auf Double Wert, Wert von R10
out	*R11*	Pointer auf Double Wert, Wert von R11
out	*R12*	Pointer auf Double Wert, Wert von R12
out	*R13*	Pointer auf Double Wert, Wert von R13
out	*R14*	Pointer auf Double Wert, Wert von R14
out	*R15*	Pointer auf Double Wert, Wert von R15

Rückgabe

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

5.14 Methoden zur Triggereinstellung

Definition der Triggereinstellungen.

Funktionen

- int __stdcall GOMDBTS2048_setTriggerSource (int handle, int value)
- int __stdcall GOMDBTS2048_getTriggerSource (int handle, int *value)
- int __stdcall GOMDBTS2048_setTriggerInternalLevels (int handle, int lightValueTrigger, int lightValueMax)
- int __stdcall GOMDBTS2048_setTriggerMode (int handle, int value)
- int __stdcall GOMDBTS2048_getTriggerMode (int handle, int *mode)
- int __stdcall GOMDBTS2048_setTriggerLevel (int handle, int level)
- int __stdcall GOMDBTS2048_getTriggerLevel (int handle, int *level)
- int __stdcall GOMDBTS2048_setTriggerInput (int handle, int input)
- int __stdcall GOMDBTS2048_getTriggerInput (int handle, int *input)
- int __stdcall GOMDBTS2048_isMeasurementFinished (int handle, bool *value)
- int __stdcall GOMDBTS2048_setTriggerTimeoutInMs (int handle, int value)
- int __stdcall GOMDBTS2048_getTriggerTimeoutInMs (int handle, int *value)
- int __stdcall GOMDBTS2048_setOut1LowDuringMeasurement (int handle, bool value)
- int __stdcall GOMDBTS2048_getOut1LowDuringMeasurement (int handle, bool *value)
- int __stdcall GOMDBTS2048_setTriggerDelay (int handle, int timeInMs)
- int __stdcall GOMDBTS2048_getTriggerDelay (int handle, int *value)

5.14.1 Ausführliche Beschreibung

5.14.2 C++ Aufrufbeispiel

Geräte soll auf high Pegel reagieren.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                    //initialization
GOMDBTS2048_setTriggerLevel(handle, 1);
GOMDBTS2048_setTriggerMode(handle, 0);
GOMDBTS2048_releaseHandle(handle);                       //release handle
```

5.14.3 Dokumentation der Funktionen

5.14.3.1 GOMDBTS2048_setTriggerSource()

```
int __stdcall GOMDBTS2048_setTriggerSource (
            int handle,
            int value )
```

Mit dieser Methode wird definiert ob das Messgerät über den externen Trigger oder über das Eintreffen des zu messenden Signals (intern) getriggert werden soll. Der Wert „0“ definiert den externen Trigger. Der Wert „1“ definiert den internen Trigger.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integerwert, der die gewünschte Triggerquelle beinhaltet: • 0: Externer Trigger • 1: Interner Trigger

Rückgabe

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

5.14.3.2 GOMDBTS2048_getTriggerSource()

```
int __stdcall GOMDBTS2048_getTriggerSource (
            int handle,
            int * value )
```

Diese Methode ermittelt die Einstellung der Triggerquelle, d.h. ob das Gerät aufgrund eines externen Triggersignals oder aufgrund des Auftretens eines bestimmten Pegels an der integralen Messeinheit getriggert werden soll.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integerwert, der die definierte Triggerquelle beinhaltetL: • 0: Externer Trigger • 1: Interner Trigger

Rückgabe

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

5.14.3.3 GOMDBTS2048_setTriggerInternalLevels()

```
int __stdcall GOMDBTS2048_setTriggerInternalLevels (
            int handle,
            int lightValueTrigger,
            int lightValueMax )
```

Mit dieser Methode werden die Levels für den internen Trigger definiert. Die Methode nimmt als Parameter den Lichtwert entgegen, bei dem die Messung ausgelöst werden soll, als auch den maximal zu erwartenden Lichtwert. Der Lichtwert bezieht sich dabei immer auf die gerade aktuell eingestellte Kalibrierung. Getriggerte Messungen sollten grundsätzlich ohne „Autorange“ betrieben werden. So schaltet diese Methode sowohl den Autorange aus, und stellt den geeigneten Range bzgl. des maximal zu erwartenden Lichtwerts ein. Der Triggerlichtwert darf 1% des maximalen zulässigen Lichtwerts nicht unterschreiten.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*lightValueTrigger*	Integerwert, der den Lichtwert enthält bei dem die Messung ausgelöst wird.
in	*lightValueMax*	Integerwert, der den maximal zu erwartenden Lichtwert enthält.

Rückgabe

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

5.14.3.4 GOMDBTS2048_setTriggerMode()

```
int __stdcall GOMDBTS2048_setTriggerMode (
            int handle,
            int value )
```

Mit dieser Methode wird definiert ob das Messgerät auf Flanken oder auf Pegel reagieren soll. Zusätzlich muss über die Methode „setTriggerLevel“ definiert werden, ob das Messgerät bei Erkennung einer Flanke (fallend bzw. steigend) oder eines festen Pegels (low bzw. high) schalten soll.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integerwert, der den gewünschten Trigger Modus beinhaltet: • 0: Pegel • 1: Flanke

Rückgabe

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

5.14.3.5 GOMDBTS2048_getTriggerMode()

```
int __stdcall GOMDBTS2048_getTriggerMode (
            int handle,
            int * mode )
```

Diese Methode ermittelt die Einstellung des Trigger Modus, d.h. ob das Gerät auf Vorhandensein einer Flanke oder eines festen Pegels am Triggereingang reagieren soll.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*mode*	Pointer auf Integerwert, der den gewünschten Trigger Modus beinhaltet • 0: Pegel • 1: Flanke

Rückgabe

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

5.14.3.6 GOMDBTS2048_setTriggerLevel()

```
int __stdcall GOMDBTS2048_setTriggerLevel (
            int handle,
            int level )
```

Mit dieser Methode wird definiert ob das Messgerät bei abfallender Flanke bzw. low Pegel am Triggereingang reagieren soll oder bei steigender Flanke bzw. high Pegel. Diese Methode muss immer im Zusammenhang von „setTriggerMode“ betrachtet werden, wo Flanke oder Pegel als Entscheidungsmerkmal gesetzt werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*level*	Doublewert, der den gewünschten Level beinhaltet • 0: Fallende Flanke oder low Pegel • 1: Steigende Flanke oder high Pegel

Rückgabe

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

5.14.3.7 GOMDBTS2048_getTriggerLevel()

```
int __stdcall GOMDBTS2048_getTriggerLevel (
            int handle,
            int * level )
```

Diese Methode ermittelt die Einstellung des Trigger Levels, d.h. ob das Gerät auf Vorhandensein einer fallenden Flanke/ low Pegels oder einer steigenden Flanke / high Pegel am Triggereingang reagieren soll.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*level*	Pointer auf Doublewert, der den gewünschten Trigger Modus beinhaltet • 0: Fallend bzw. low • 1: Steigend bzw. high

Rückgabe

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

5.14.3.8 GOMDBTS2048_setTriggerInput()

```
int __stdcall GOMDBTS2048_setTriggerInput (
             int handle,
             int input )
```

Mit dieser Methode wird definiert auf welchen Triggereingang das Messgerät reagieren soll.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*input*	Integerwert, der den gewünschten Triggereingang beinhaltet: • 1: Triggereingang 1 • 2: Triggereingang 2 • 3: Triggereingang 3

Rückgabe

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

5.14.3.9 GOMDBTS2048_getTriggerInput()

```
int __stdcall GOMDBTS2048_getTriggerInput (
             int handle,
             int * input )
```

Diese Methode ermittelt die Einstellung, auf welchen Triggereingang das BTS2048 reagieren soll.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*input*	Pointer auf Integerwert, der den selektierten Triggerinput beinhaltet: • 1: Triggereingang 1 • 2: Triggereingang 2 • 3: Triggereingang 3

Rückgabe

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

5.14.3.10 GOMDBTS2048_isMeasurementFinished()

```
int __stdcall GOMDBTS2048_isMeasurementFinished (
```

```
            int handle,
            bool * value )
```

Diese Methode prüft, ob eine getriggerte Messung bereits fertiggestellt ist. Bei einer getriggerten Messung muss diese Methode zuerst „true“ zurückliefern, bevor eine andere Methode aufgerufen werden darf.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • true: Messung fertig • false: Messung noch nicht erfolgt

Rückgabe

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

5.14.3.11 GOMDBTS2048_setTriggerTimeoutInMs()

```
int __stdcall GOMDBTS2048_setTriggerTimeoutInMs (
            int handle,
            int value )
```

Mit dieser Methode setzt man die Zeit, die gewartet wird bis ein Trigger für eine getriggerte Messung eintrifft. Wenn diese Zeit abgelaufen ist, wird die getriggerte Messung abgebrochen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert; die zu wartende Zeit in Millisekunden [ms].

Rückgabe

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

5.14.3.12 GOMDBTS2048_getTriggerTimeoutInMs()

```
int __stdcall GOMDBTS2048_getTriggerTimeoutInMs (
            int handle,
            int * value )
```

Liefert die zuvor gesetzte „Timeout“-Zeit für getriggerte Messungen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer Wert; die zu wartende Zeit in Millisekunden [ms].

Rückgabe

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

5.14.3.13 GOMDBTS2048_setOut1LowDuringMeasurement()

```
int __stdcall GOMDBTS2048_setOut1LowDuringMeasurement (
            int handle,
            bool value )
```

Diese Methode aktiviert die Funktionalität, dass während der Integrationszeit der spektralen Einheit der Output 1 - Pin des 18 poligen D-Sub Buchse auf „Low“ gesetzt wird. So kann ein extern getriggertes Gerät den Beginn und das Ende der Integrationszeit des Messprozesses feststellen.
Die weitergehende Verarbeitung (Berechnungen, . . .) sind zu diesem Zeitpunkt noch nicht abgeschlossen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • false: Output 1 wird nicht auf Low gesetzt. Keine Funktionalität des output 1 Pin. • true: Output 1 wird während Messung auf Low gesetzt

Rückgabe

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

5.14.3.14 GOMDBTS2048_getOut1LowDuringMeasurement()

```
int __stdcall GOMDBTS2048_getOut1LowDuringMeasurement (
            int handle,
            bool * value )
```

Diese Methode gibt zurück, ob der Output 1 während der Messung auf „Low“ gesetzt wird.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert; enthält nach Rücksprung den Status: • false: Output 1 wird nicht auf Low gesetzt • true: Output 1 wird während der Messung auf Low gesetzt

Rückgabe

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

5.14.3.15 GOMDBTS2048_setTriggerDelay()

```
int __stdcall GOMDBTS2048_setTriggerDelay (
            int handle,
            int timeInMs )
```

Mit dieser Methode legt man die Triggerverzögerung fest. Diese Verzögert eine getriggerte Messung nach Eingang des Triggersignals um den angegebenen Wert.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*timeInMs*	Integer Wert der Triggerverzögerung in ms.

Rückgabe

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

5.14.3.16 GOMDBTS2048_getTriggerDelay()

```
int __stdcall GOMDBTS2048_getTriggerDelay (
            int handle,
            int * value )
```

Liefert die Triggerverzögerung die im Gerät eingestellt wurde.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer Wert, enthält nach Rücksprung die gesetzte Triggerverzögerung in ms.

Rückgabe

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

5.15 Methoden für die Substitutionskorrektur DUT (Device Under Test)

Die folgenden Methoden dienen der Substitutionskorrektur.

Funktionen

- int __stdcall GOMDBTS2048_substitutionEnableCorrection (int handle, bool active)
- int __stdcall GOMDBTS2048_substitutionIsEnabledCorrection (int handle, bool *active)
- int __stdcall GOMDBTS2048_substitutionMeasurementWithoutTestDevice (int handle, bool isExternal↩ Sphere, double *saturation, double *counts, double *current)
- int __stdcall GOMDBTS2048_substitutionMeasurementWithTestDevice (int handle, bool isExternalSphere, double *saturation, double *counts, double *current)
- int __stdcall GOMDBTS2048_substitutionSetIntegrationTimeInUs (int handle, int timeInus)
- int __stdcall GOMDBTS2048_substitutionGetIntegrationTimeInUs (int handle, int *timeInus)
- int __stdcall GOMDBTS2048_substitutionSetDynamicTimeMode (int handle, bool value)
- int __stdcall GOMDBTS2048_substitutionGetDynamicTimeMode (int handle, bool *value)
- int __stdcall GOMDBTS2048_substitutionSetHighResolutionMode (int handle, bool value)
- int __stdcall GOMDBTS2048_substitutionGetHighResolutionMode (int handle, bool *value)
- int __stdcall GOMDBTS2048_substitutionSaveFactors (int handle, char *absoluteFileName)
- int __stdcall GOMDBTS2048_substitutionLoadFactors (int handle, char *absoluteFileName)
- int __stdcall GOMDBTS2048_substitutionGetLoadedFilename (int handle, char *absoluteFileName)
- int __stdcall GOMDBTS2048_substitutionGetSpectralFactor (int handle, int pixelNumber, double *factor)
- int __stdcall GOMDBTS2048_substitutionGetSpectralFactors (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionGetPresetSpectralFactors (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionGetIntegralFactor (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionGetPresetIntegralFactor (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionSetComment (int handle, char *comment)
- int __stdcall GOMDBTS2048_substitutionGetComment (int handle, char *comment)
- int __stdcall GOMDBTS2048_substitutionGetDateTime (int handle, int *day, int *month, int *year, int *hh, int *mm, int *ss)

5.15.1 Ausführliche Beschreibung

Die folgenden Methoden dienen der Substitutionskorrektur. Diese sind nur bei der Verwendung von Ulbrichtschen Kugeln notwendig. Diese Kugeln werden im leeren Zustand kalibriert. Durch Einbringung eines Testobjekts inklusive Halterung verändert sich die Eigenschaften der Kugel, Messergebnisse werden verfälscht. Daher muss für unterschiedliche Testobjekte vor Durchführung einer Messung zuerst die Substitutionskorrektur durchgeführt werden. Die Substitutionsfaktorermittlung muss jedoch nur einmal für die entsprechenden Testobjekte durchgeführt werden, da die Substitutionsfaktoren gespeichert und wieder reaktiviert werden können.
Eine zweite Verfälschung von Messergebnissen ergibt sich durch Änderungen an der Kugelgeometrie. Für Änderungen an der Kugelgeometrie, muss ebenso eine Substitutionskorrektur durchgeführt werden. Es gibt zwei Methodengruppen für die beiden Substitutionsarten (Substitutionskorrektur DUT und Substitutionskorrektur Geometrie).

5.15.2 C++ Aufrufbeispiel

Setze 5µs Integrationszeit für das Spektrometer.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                                    //
      initialization
GOMDBTS2048_substitutionSetIntegrationTimeInUs(handle, 5);
GOMDBTS2048_releaseHandle(handle);                                       //release
       handle
```

5.15.3 Dokumentation der Funktionen

5.15.3.1 GOMDBTS2048_substitutionEnableCorrection()

```
int __stdcall GOMDBTS2048_substitutionEnableCorrection (
              int handle,
              bool active )
```

Diese Methode schaltet die Substitutionskorrektur für ein Messobject (DUT = device under test) ein oder aus. Substitutionskorrektur ist ein notwendiger Schritt, wenn Sie Messungen mit einer Ulbricht'schen Kugel durchführen. Die Kalibrierwerte, welche für eine Messung des Lichtstroms im BTS2048 abgelegt sind, werden immer mit einer leeren Kugel ermittelt. Sobald sich ein Messobjekt in der Kugel befindet, werden die Eigenschaften der Kugel durch das Messobjekt verändert. Diese Veränderung, die zu einem Messfehler führen würde, muss korrigiert werden. Dies wird mit Hilfe der Substitutionskorrektur erledigt.
Damit diese Methode wirksam wird, müssen vorher bereits Substitutionsfaktoren für das Messobjekt ermittelt worden sein. Dies geschieht mit den Methoden "substitutionMeasurementWithoutDevice" and "substitution↩ MeasurementWithDevice". Diese Faktoren können gespeichert ("substitutionLoadFactors") und später auch wieder geladen ("substitutionLoadFactors") werden. Initial sind alle Faktoren mit "1.0" belegt. Dies ist der neutrale Faktor, der zu keiner Korrektur führt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*active*	Booleanwert, der bestimmt, ob die Substitutionskorrektur aktiviert wird oder nicht: • true: Substitutionskorrektur wird aktiviert • false: Substitutionskorrektur wird deaktiviert

Rückgabe

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

5.15.3.2 GOMDBTS2048_substitutionIsEnabledCorrection()

```
int __stdcall GOMDBTS2048_substitutionIsEnabledCorrection (
              int handle,
              bool * active )
```

Prüft, ob die Substitutionskorrektur eingeschaltet ist, oder nicht.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*active*	Booleanwert, enthält die Information, ob Susbtitutionskorrektur aktiv ist oder nicht: • true: Substitutionskorrektur aktiviert • false: Substitutionskorrektur deaktiviert

Rückgabe

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

5.15.3.3 GOMDBTS2048_substitutionMeasurementWithoutTestDevice()

```
int __stdcall GOMDBTS2048_substitutionMeasurementWithoutTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

Dieses ist immer die erste Methode, die verwendet werden muss um eine Substitution für ein bestimmtes Messobjekt in einer Ulbricht‘schen Kugel zu ermitteln. Der erste Schritt ist immer die Messung der leeren Kugel. Danach kann die Substitutionsermittlung mit “substitutionMeasurementWithTestDevice“ fortgeführt werden.
Zur Vermessung der leeren Kugel muss zuerst die Hilfslampe eingeschaltet werden, die mit Ihrer Kugel ausgeliefert wurde. Lassen Sie die Hilfslampe für die vorgegebene Zeit einbrennen um ein möglichst stabiles Signal zu haben. Versuchen sie, die Integrationszeit so zu ermitteln, dass eine Sättigung von 54% bis 95% erreicht werden. Bei kleinerer Aussteuerung können die ermittelten Substitutionsfaktoren ungenau werden. Entfernen Sie sämtliche Inhalte aus der Kugel und rufen Sie diese Methode auf.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*isExternalSphere*	Booleanwert, derzeit ohne Bedeutung!
out	*saturation*	Pointer auf Doublewert, enthält die Aussteuerung der durchgeführten Substitutionsmessung in [%].
out	*counts*	Array mit Doublewerten, Größe 2048 Elemente, enthält die Rohwerte (Counts) der spektralen Messung.
out	*current*	Pointer auf Doublewert, enthält den Rohwert (Stromstärke) des integralen Sensors.

Rückgabe

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

5.15.3.4 GOMDBTS2048_substitutionMeasurementWithTestDevice()

```
int __stdcall GOMDBTS2048_substitutionMeasurementWithTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

Dieses ist die zweite Methode, die zur Ermittlung von Substitutionsfaktoren aufgerufen werden muss. Schalten Sie Ihre Hilfslampe an und warten Sie die vom Hersteller vorgegebene Einbrennzeit ab, um ein stabiles Signal zu erhalten.
Verwenden Sie bei Messung der mit dem Messobjekt bestückten Kugel immer die gleiche Integrationszeit, die Sie bereits bei der Messung der leeren Kugel verwendet hatten.

Falls die Aussteuerung außerhalb des Bereichs von 54% bis 95% liegen sollte, so wiederholen Sie bitte zuerst die Messung der leeren Kugel mit geänderter Integrationszeit.
Es gibt Messobjekte, für die es nicht möglich ist bei beiden Messungen (leere und bestückte Kugel) in den optimalen Aussteuerungsbereich zu gelangen. Versuchen Sie in diesem Fall, die für diese Situation am besten mögliche Aussteuerung zu erreichen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*isExternalSphere*	Booleanwert, derzeit ohne Bedeutung!
out	*saturation*	Pointer auf Doublewert, enthält die Aussteuerung der durchgeführten Substitutionsmessung in [%].
out	*counts*	Array mit Doublewerten, Größe 2048 Elemente, enthält die Rohwerte (Counts) der spektralen Messung.
out	*current*	Pointer auf Doublewert, enthält den Rohwert (Stromstärke) des integralen Sensors.

Rückgabe

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

5.15.3.5 GOMDBTS2048_substitutionSetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionSetIntegrationTimeInUs (
            int handle,
            int timeInus )
```

Mit dieser Methode kann die Integrationszeit des Spektrometers definiert werden, die für die Substitutionsmessung verwendet wird. Die Integrationszeiten müssen in der Einheit µs an die Methode übergeben werden. Wertebereich: 2 – 4000000 -> 2µs bis 4sec. Falls Sie ein Gerät mit Kühlung besitzen (Kühlung muss eingeschaltet sein), dann sind Werte bis 60000000 µs, also 60sec. zulässig. Wenn die Integrationszeit zu lange gewählt wurde, dann übersteuert das Spektrometer und die Messergebnisse können unbrauchbar sein.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*timeInus*	Integer Wert, die Integrationszeit in µSekunden [µs].

Rückgabe

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

5.15.3.6 GOMDBTS2048_substitutionGetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionGetIntegrationTimeInUs (
            int handle,
            int * timeInus )
```

Diese Methode liefert die für das Spektrometer zuletzt gesetzte Integrationszeit in der Einheit µs zurück.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*timeInus*	Pointer auf Integer; enthält nach Rücksprung die Integrationszeit in [µs].

Rückgabe

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

5.15.3.7 GOMDBTS2048_substitutionSetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionSetDynamicTimeMode (
            int handle,
            bool value )
```

Mit dieser Methode wird die dynamische Integrationzeitanpassung für die Substitution aktiviert.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true: Dynamische Integrationszeitermittlung aktiv • false: Dynamische Integrationszeitermittlung nicht aktiv

Rückgabe

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

5.15.3.8 GOMDBTS2048_substitutionGetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionGetDynamicTimeMode (
            int handle,
            bool * value )
```

Diese Methode liefert zurück ob die dynamische Integrationszeit-Bestimmung für die Substitution aktiviert ist

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean: • true: Dynamische Integrationszeitermittlung ist aktiv • false: Dynamische Integrationszeitermittlung ist nicht aktiv

Rückgabe

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

5.15.3.9 GOMDBTS2048_substitutionSetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionSetHighResolutionMode (
            int handle,
            bool value )
```

Diese Methode schaltet den hochauflösenden Messmodus für die Substitutionsmessung an und aus. Dabei wird das Spektrum aus mehereren Messungen mit unterschiedlichen Integrationszeiten zusammengesetzt.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • false: Hochauflösende Messung deaktiviert • true: Hochauflösende Messung aktiviert

Rückgabe

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

5.15.3.10 GOMDBTS2048_substitutionGetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionGetHighResolutionMode (
            int handle,
            bool * value )
```

Diese Methode gibt den Status der hochauflösenden Substitutionsmessung zurück.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • false: Hochauflösende Messung deaktiviert • true: Hochauflösende Messung aktiviert

Rückgabe

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

5.15.3.11 GOMDBTS2048_substitutionSaveFactors()

```
int __stdcall GOMDBTS2048_substitutionSaveFactors (
             int handle,
             char * absoluteFileName )
```

Aktuell ermittelte Substitutionsfaktoren werden unter dem definierten Dateinamen abgespeichert. Dateiname muss mit komplettem Pfad angegeben werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*absoluteFileName*	Nullterminierter String, enthält den kompletten Pfad auf die zu speichernde Datei.

Rückgabe

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

5.15.3.12 GOMDBTS2048_substitutionLoadFactors()

```
int __stdcall GOMDBTS2048_substitutionLoadFactors (
             int handle,
             char * absoluteFileName )
```

Ladet den zuvor gemessenen und abgespeicherten (GOMDBTS2048_substitutionSaveFactors()) Substitutionsfaktor in das BTS2048. Dieser Faktor wird benutzt, wenn "substitutuion correction" eingeschalten ist. Dies ist mit substitutionEnableCorrection() möglich.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*absoluteFileName*	Nullterminierter String, enthält den kompletten Pfad auf die zu ladende Datei.

Rückgabe

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

5.15.3.13 GOMDBTS2048_substitutionGetLoadedFilename()

```
int __stdcall GOMDBTS2048_substitutionGetLoadedFilename (
             int handle,
             char * absoluteFileName )
```

Liefert den Dateinamen mit komplettem Pfad zu der Datei, aus welcher die aktuellen Substitutionsfaktoren geladen wurden. Wurden die Substitutionsfaktoren nicht aus einer Datei geladen wird ein Leerstring zurückgeschickt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*absoluteFileName*	Nullterminierter String, muss mit 2048 Bytes allokiert werden.

Rückgabe

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

5.15.3.14 GOMDBTS2048_substitutionGetSpectralFactor()

```
int __stdcall GOMDBTS2048_substitutionGetSpectralFactor (
            int handle,
            int pixelNumber,
            double * factor )
```

Diese Methode liefert den aktuellen Substitutionsfaktor für ein spezifisches Pixel des Spektrometers. Dieser Faktor findet bei der aktuellen Messung Verwendung. Falls die Substitutionskorrektur ausgeschaltet ist, wird hier immer der Wert 1.0 zurückgegeben. Wenn der voreingestellte Substitutionsfaktor ermittelt werden soll, der aktuell vorhanden ist (unabhängig davon, ob Substitutionskorrektur aktiv oder nicht), dann sollte die Methode substitutionGetPreset↩ SpectralFactors verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*pixelNumber*	Integerwert, enthält die gewünschte Pixelnummer, Wertebereich: 0 - 2047.
out	*factor*	Pointer auf Doublewert, enthält den Subsitutionsfaktor für das spezifizierte Pixel.

Rückgabe

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

5.15.3.15 GOMDBTS2048_substitutionGetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGetSpectralFactors (
            int handle,
            double * factor )
```

Diese Methode liefert alle 2048 aktuellen Substitutionsfaktoren für ein spezifisches Pixel des Spektrometers. Diesee Faktoren finden bei der aktuellen Messung Verwendung. Falls die Substitutionskorrektur ausgeschaltet ist, wird hier immer der Wert 1.0 zurückgegeben. Wenn die voreingestellten Substitutionsfaktoren ermittelt werden sollen, die aktuell vorhanden sind (unabhängig davon, ob Substitutionskorrektur aktiv oder nicht), dann sollte die Methode substitutionGetPresetSpectralFactors verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf ersten Wert eines Double Array, enthält die Subsitutionsfaktor für alle Pixel; das Array muss für 2048 double Werte allokiert werden.

Rückgabe

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

5.15.3.16 GOMDBTS2048_substitutionGetPresetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGetPresetSpectralFactors (
            int handle,
            double * factor )
```

Diese Methode liefert alle voreingestellten Substitutionsfaktoren für den spektralen Sensor, unabhängig davon, ob die Substitutionskorrektur eingeschaltet ist, oder nicht. Diese Faktoren finden nur dann Verwendung, wenn die Substitutionskorrektur aktiviert ist. Für die bei der Messung tatsächlich verwendeten Korrekturfaktoren sollte die Methode substitutionGetSpectralFactor verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf das erste Element eines Double Array, enthält nach Rücksprung die voreingestellten Korrekturfaktoren. Das Array benltigt Platz für 2048 double Werte

Rückgabe

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

5.15.3.17 GOMDBTS2048_substitutionGetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGetIntegralFactor (
            int handle,
            double * factor )
```

Diese Methode liefert den aktuellen Substitutionsfaktor für den integralen Sensor. Dieser Faktor findet bei der aktuellen Messung Verwendung. Falls die Substitutionskorrektur ausgeschaltet ist, wird hier immer der Wert 1.0 zurückgegeben. Wenn der voreingestellte Substitutionsfaktor ermittelt werden soll, der aktuell vorhanden ist (unabhängig davon, ob Substitutionskorrektur aktiv oder nicht), dann sollte die Methode substitutionGetPresetIntegralFactor verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf Doublewert, enthält den Substitutionsfaktor.

Rückgabe

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

5.15.3.18 GOMDBTS2048_substitutionGetPresetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGetPresetIntegralFactor (
            int handle,
            double * factor )
```

Diese Methode liefert den voreingestellten Substitutionsfaktor für den integralen Sensor, unabhängig davon, ob die Substitutionskorrektur eingeschaltet ist, oder nicht. Dieser Faktor findet nur dann Verwendung, wenn die Substitutionskorrektur aktiviert ist. Für den bei der Messung tatsächlich verwendeten Korrekturfaktor sollte die Methode substitutionGetIntegralFactor verwendet werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf Doublewert, enthält den Substitutionsfaktor.

Rückgabe

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

5.15.3.19 GOMDBTS2048_substitutionSetComment()

```
int __stdcall GOMDBTS2048_substitutionSetComment (
            int handle,
            char * comment )
```

Diese Methode setzt einen Kommentar, der die aktuelle Substitution näher beschreibt. Sie sollte vor dem Speichern der aktuellen Substitutionskorrektur aufgerufen werden, da dieser Kommentar mit abgespeichert wird.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*comment*	nullterminierter String, der den Kommentar beinhaltet; maximal zulässige Länge inklusive Terminator: 1024 Bytes.

Rückgabe

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

5.15.3.20 GOMDBTS2048_substitutionGetComment()

```
int __stdcall GOMDBTS2048_substitutionGetComment (
            int handle,
            char * comment )
```

Diese Methode liefert den Kommentar, der durch die Methode substitutionSetComment gesetzt wurde.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*comment*	nullterminierter String, muss mit 1024 Bytes allokiert werden; enthält nach Rücksprung den Kommentar.

Rückgabe

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

5.15.3.21 GOMDBTS2048_substitutionGetDateTime()

```
int __stdcall GOMDBTS2048_substitutionGetDateTime (
              int handle,
              int * day,
              int * month,
              int * year,
              int * hh,
              int * mm,
              int * ss )
```

Diese Methode liefert das Datum und die Uhrzeit, zu der die aktuelle Substitutitonskorrektur gespeichert wurde.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*day*	Pointer auf Integer, enthält nach Rücksprung den Tag des Datums
out	*month*	Pointer auf Integer, enthält nach Rücksprung den Monat des Datums
out	*year*	Pointer auf Integer, enthält nach Rücksprung das Jahr des Datums
out	*hh*	Pointer auf Integer, enthält nach Rücksprung die Stunden der Uhrzeit
out	*mm*	Pointer auf Integer, enthält nach Rücksprung die Minuten der Uhrzeit
out	*ss*	Pointer auf Integer, enthält nach Rücksprung die Sekunden der Uhrzeit

Rückgabe

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

5.16 Methoden für die Substitutionskorrektur Geometrie

Die folgenden Methoden dienen der Substitutionskorrektur.

Funktionen

- int __stdcall GOMDBTS2048_substitutionGeoEnableCorrection (int handle, bool active)
- int __stdcall GOMDBTS2048_substitutionGeoIsEnabledCorrection (int handle, bool *active)
- int __stdcall GOMDBTS2048_substitutionGeoMeasurementWithoutTestDevice (int handle, bool isExternal↩ Sphere, double *saturation, double *counts, double *current)
- int __stdcall GOMDBTS2048_substitutionGeoMeasurementWithTestDevice (int handle, bool isExternal↩ Sphere, double *saturation, double *counts, double *current)
- int __stdcall GOMDBTS2048_substitutionGeoSetIntegrationTimeInUs (int handle, int timeInus)
- int __stdcall GOMDBTS2048_substitutionGeoGetIntegrationTimeInUs (int handle, int *timeInus)
- int __stdcall GOMDBTS2048_substitutionGeoSetDynamicTimeMode (int handle, bool value)
- int __stdcall GOMDBTS2048_substitutionGeoGetDynamicTimeMode (int handle, bool *value)
- int __stdcall GOMDBTS2048_substitutionGeoSetHighResolutionMode (int handle, bool value)
- int __stdcall GOMDBTS2048_substitutionGeoGetHighResolutionMode (int handle, bool *value)
- int __stdcall GOMDBTS2048_substitutionGeoSaveFactors (int handle, char *absoluteFileName)
- int __stdcall GOMDBTS2048_substitutionGeoLoadFactors (int handle, char *absoluteFileName)
- int __stdcall GOMDBTS2048_substitutionGeoGetLoadedFilename (int handle, char *absoluteFileName)
- int __stdcall GOMDBTS2048_substitutionGeoGetSpectralFactor (int handle, int pixelNumber, double *factor)
- int __stdcall GOMDBTS2048_substitutionGeoGetSpectralFactors (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionGeoGetPresetSpectralFactors (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionGeoGetIntegralFactor (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionGeoGetPresetIntegralFactor (int handle, double *factor)
- int __stdcall GOMDBTS2048_substitutionGeoSetComment (int handle, char *comment)
- int __stdcall GOMDBTS2048_substitutionGeoGetComment (int handle, char *comment)
- int __stdcall GOMDBTS2048_substitutionGeoGetDateTime (int handle, int *day, int *month, int *year, int *hh, int *mm, int *ss)

5.16.1 Ausführliche Beschreibung

Die Sbstiutionsorrektur Geometire ist für jegliche Änderung gedacht, die an der Kugel-Geometrie durchgeführt werden. Sie funktioniert identisch zu der Substitutionskorrtur DUT.

5.16.2 C++ Aufrufbeispiel

Setze 5µs Integrationszeit für das Spektrometer

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                                    //
      initialization
GOMDBTS2048_substitutionGeoSetIntegrationTimeInUs(handle,
      5);
GOMDBTS2048_releaseHandle(handle);                                       //release
       handle
```

5.16.3 Dokumentation der Funktionen

5.16.3.1 GOMDBTS2048_substitutionGeoEnableCorrection()

```
int __stdcall GOMDBTS2048_substitutionGeoEnableCorrection (
            int handle,
            bool active )
```

Diese Methode schaltet die Substitutionskorrektur der Geometrie ein oder aus. Substitutionskorrektur ist ein notwendiger Schritt, wenn Sie Messungen mit einer Ulbricht'schen Kugel durchführen. Die Kalibrierwerte, welche für eine Messung des Lichtstroms im BTS2048 abgelegt sind, werden immer mit einer leeren Kugel ermittelt. Sobald sich die Geometrie der Kugel andert, werden die Reflexionseigenschaften der Kugel verändert. Diese Veränderung, die zu einem Messfehler führen würde, muss korrigiert werden. Dies wird mit Hilfe der Substitutionskorrektur erledigt.
Damit diese Methode wirksam wird, müssen vorher bereits Substitutionsfaktoren für das Messobjekt ermittelt worden sein. Dies geschieht mit den Methoden "substitutionMeasurementWithoutDevice" and "substitution↩ MeasurementWithDevice". Diese Faktoren können gespeichert ("substitutionLoadFactors") und später auch wieder geladen ("substitutionLoadFactors") werden. Initial sind alle Faktoren mit "1.0" belegt. Dies ist der neutrale Faktor, der zu keiner Korrektur führt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*active*	Booleanwert, der bestimmt, ob die Substitutionskorrektur aktiviert wird oder nicht. • true: Substitutionskorrektur wird aktiviert • false: Substitutionskorrektur wird deaktiviert

Rückgabe

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

5.16.3.2 GOMDBTS2048_substitutionGeoIsEnabledCorrection()

```
int __stdcall GOMDBTS2048_substitutionGeoIsEnabledCorrection (
            int handle,
            bool * active )
```

Prüft, ob die Substitutionskorrektur der Geometrie eingeschaltet ist, oder nicht.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*active*	Pointer auf Booleanwert, enthält die Information, ob Susbtitutionskorrektur aktiv ist oder nicht: • true: Substitutionskorrektur aktiv • false: Substitutionskorrektur nicht aktiv

Rückgabe

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

5.16.3.3 GOMDBTS2048_substitutionGeoMeasurementWithoutTestDevice()

```
int __stdcall GOMDBTS2048_substitutionGeoMeasurementWithoutTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

Dieses ist immer die erste Methode, die verwendet werden muss um eine Substitution für ein bestimmtes Messobjekt in einer Ulbricht'schen Kugel zu ermitteln. Der erste Schritt ist immer die Messung der leeren Kugel. Danach kann die Substitutionsermittlung mit “substitutionGeoMeasurementWithTestDevice“ fortgeführt werden.
Zur Vermessung der leeren Kugel muss zuerst die Hilfslampe eingeschaltet werden, die mit Ihrer Kugel ausgeliefert wurde. Lassen Sie die Hilfslampe für die vorgegebene Zeit einbrennen um ein möglichst stabiles Signal zu haben. Versuchen sie, die Integrationszeit so zu ermitteln, dass eine Sättigung von 54% bis 95% erreicht werden. Bei kleinerer Aussteuerung können die ermittelten Substitutionsfaktoren ungenau werden. Entfernen Sie sämtliche Inhalte aus der Kugel und rufen Sie diese Methode auf.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*isExternalSphere*	Booleanwert, derzeit ohne Bedeutung .
out	*saturation*	Pointer auf Doublewert, enthält die Aussteuerung der durchgeführten Substitutionsmessung in [%].
out	*counts*	Array mit Doublewerten, Größe 2048 Elemente, enthält die Rohwerte (Counts) der spektralen Messung.
out	*current*	Pointer auf Doublewert, enthält den Rohwert (Stromstärke) des integralen Sensors.

Rückgabe

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

5.16.3.4 GOMDBTS2048_substitutionGeoMeasurementWithTestDevice()

```
int __stdcall GOMDBTS2048_substitutionGeoMeasurementWithTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

Dieses ist die zweite Methode, die zur Ermittlung von Substitutionsfaktoren aufgerufen werden muss. Schalten Sie Ihre Hilfslampe an und warten Sie die vom Hersteller vorgegebene Einbrennzeit ab, um ein stabiles Signal zu erhalten.
Verwenden Sie bei Messung der mit dem Messobjekt bestückten Kugel immer die gleiche Integrationszeit, die Sie bereits bei der Messung der leeren Kugel verwendet hatten. Falls die Aussteuerung außerhalb des Bereichs von

54% bis 95% liegen sollte, so wiederholen Sie bitte zuerst die Messung der leeren Kugel mit geänderter Integrationszeit.
Es gibt Messobjekte, für die es nicht möglich ist bei beiden Messungen (leere und bestückte Kugel) in den optimalen Aussteuerungsbereich zu gelangen. Versuchen Sie in diesem Fall, die für diese Situation am besten mögliche Aussteuerung zu erreichen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*isExternalSphere*	Booleanwert, derzeit ohne Bedeutung.
out	*saturation*	Pointer auf Doublewert, enthält die Aussteuerung der durchgeführten Substitutionsmessung in [%].
out	*counts*	Array mit Doublewerten, Größe 2048 Elemente, enthält die Rohwerte (Counts) der spektralen Messung.
out	*current*	Pointer auf Doublewert, enthält den Rohwert (Stromstärke) des integralen Sensors.

Rückgabe

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

5.16.3.5 GOMDBTS2048_substitutionGeoSetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionGeoSetIntegrationTimeInUs (
            int handle,
            int timeInus )
```

Mit dieser Methode kann die Integrationszeit des Spektrometers definiert werden, die für die Substitutionsmessung verwendet wird. Die Integrationszeiten müssen in der Einheit µs an die Methode übergeben werden. Wertebereich: 2 – 4000000 -> 2µs bis 4sec. Falls Sie ein Gerät mit Kühlung besitzen (Kühlung muss eingeschaltet sein), dann sind Werte bis 60000000 µs, also 60sec. zulässig. Wenn die Integrationszeit zu lange gewählt wurde, dann übersteuert das Spektrometer und die Messergebnisse können unbrauchbar sein.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*timeInus*	Integer Wert, die Integrationszeit in µSekunden.

Rückgabe

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

5.16.3.6 GOMDBTS2048_substitutionGeoGetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionGeoGetIntegrationTimeInUs (
            int handle,
            int * timeInus )
```

Diese Methode liefert die für das Spektrometer zuletzt gesetzte Integrationszeit in der Einheit µs zurück.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*timeInus*	Pointer auf Integer; enthält nach Rücksprung die Integrationszeit in [µs].

Rückgabe

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

5.16.3.7 GOMDBTS2048_substitutionGeoSetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionGeoSetDynamicTimeMode (
            int handle,
            bool value )
```

Mit dieser Methode wird die dynamische Integrationzeitanpassung für die Substitution aktiviert.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • true: Dynamische Integrationszeitermittlung aktiv • false: Dynamische Integrationszeitermittlung nicht aktiv

Rückgabe

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

5.16.3.8 GOMDBTS2048_substitutionGeoGetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionGeoGetDynamicTimeMode (
            int handle,
            bool * value )
```

Diese Methode liefert zurück ob die dynamische Integrationszeit-Bestimmung für die Substitution aktiviert ist

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean; enthält nach Rücksprung den „time mode“: • true: Dynamische Integrationszeitermittlung aktiv • false: Dynamische Integrationszeitermittlung nicht aktiv

Rückgabe

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

5.16.3.9 GOMDBTS2048_substitutionGeoSetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionGeoSetHighResolutionMode (
            int handle,
            bool value )
```

Diese Methode schaltet den hochauflösenden Messmodus für die Substitutionsmessung an und aus. Dabei wird das Spektrum aus mehereren Messungen mit unterschiedlichen Integrationszeiten zusammengesetzt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Boolean Wert: • false: Hochauflösende Messung deaktiviert • true: Hochauflösende Messung aktiviert

Rückgabe

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

5.16.3.10 GOMDBTS2048_substitutionGeoGetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionGeoGetHighResolutionMode (
            int handle,
            bool * value )
```

Diese Methode gibt den Status der hochauflösenden Substitutionsmessung zurück.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert: • false: Hochauflösende Messung deaktiviert • true: Hochauflösende Messung aktiviert

Rückgabe

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

5.16.3.11 GOMDBTS2048_substitutionGeoSaveFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoSaveFactors (
             int handle,
             char * absoluteFileName )
```

Aktuell ermittelte Substitutionsfaktoren werden unter dem definierten Dateinamen abgespeichert. Dateiname muss mit komplettem Pfad angegeben werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*absoluteFileName*	Nullterminierter String, kompletter Pfad zu Dateiname.

Rückgabe

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

5.16.3.12 GOMDBTS2048_substitutionGeoLoadFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoLoadFactors (
             int handle,
             char * absoluteFileName )
```

Ladet den zuvor gemessenen und abgespeicherten Substitutionsfaktor von einer Datei in das BTS2048. Dieser Faktor wird bei eingeschalteter Substitution benutzt. Mit dem Aufruf substitutionGeoEnableCorrection() kann überprüft werden, ob die Substitution eingeschalten ist.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*absoluteFileName*	Nullterminierter String, enthält den kompletten Pfad auf die zu ladende Datei.

Rückgabe

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

5.16.3.13 GOMDBTS2048_substitutionGeoGetLoadedFilename()

```
int __stdcall GOMDBTS2048_substitutionGeoGetLoadedFilename (
             int handle,
             char * absoluteFileName )
```

Liefert den Dateinamen mit komplettem Pfad zu der Datei, aus welcher die aktuellen Substitutionsfaktoren geladen wurden. Wurden die Substitutionsfaktoren nicht aus einer Datei geladen wird ein Leerstring zurückgeschickt.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*absoluteFileName*	Nullterminierter String, muss mit 2048 Bytes allokiert werden.

Rückgabe

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

5.16.3.14 GOMDBTS2048_substitutionGeoGetSpectralFactor()

```
int __stdcall GOMDBTS2048_substitutionGeoGetSpectralFactor (
            int handle,
            int pixelNumber,
            double * factor )
```

Diese Methode liefert den aktuellen Substitutionsfaktor für ein spezifisches Pixel des Spektrometers. Dieser Faktor findet bei der aktuellen Messung Verwendung. Falls die Substitutionskorrektur ausgeschaltet ist, wird hier immer der Wert 1.0 zurückgegeben. Wenn der voreingestellte Substitutionsfaktor ermittelt werden soll, der aktuell vorhanden ist (unabhängig davon, ob Substitutionskorrektur aktiv oder nicht), dann sollte die Methode substitutionGeoGet↩ PresetSpectralFactors verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*pixelNumber*	Integerwert, enthält die gewünschte Pixelnummer, Wertebereich: 0 - 2047.
out	*factor*	Pointer auf Doublewert, enthält den Substitutionsfaktor für das spezifizierte Pixel.

Rückgabe

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

5.16.3.15 GOMDBTS2048_substitutionGeoGetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoGetSpectralFactors (
            int handle,
            double * factor )
```

Diese Methode liefert alle 2048 aktuellen Substitutionsfaktoren für ein spezifisches Pixel des Spektrometers. Dieser Faktoren finden bei der aktuellen Messung Verwendung. Falls die Substitutionskorrektur ausgeschaltet ist, wird hier immer der Wert 1.0 zurückgegeben. Wenn die voreingestellten Substitutionsfaktoren ermittelt werden sollen, die aktuell vorhanden sind (unabhängig davon, ob Substitutionskorrektur aktiv oder nicht), dann sollte die Methode substitutionGeoGetPresetSpectralFactors verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf ersten Wert eines Double Array, enthält die Subsitutionsfaktor für alle Pixel; das Array muss für 2048 double Werte allokiert werden.

Rückgabe

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

5.16.3.16 GOMDBTS2048_substitutionGeoGetPresetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoGetPresetSpectralFactors (
            int handle,
            double * factor )
```

Diese Methode liefert alle voreingestellten Substitutionsfaktoren für den spektralen Sensor, unabhängig davon, ob die Substitutionskorrektur eingeschaltet ist, oder nicht. Diese Faktoren finden nur dann Verwendung, wenn die Substitutionskorrektur aktiviert ist. Für die bei der Messung tatsächlich verwendeten Korrekturfaktoren sollte die Methode substitutionGeoGetSpectralFactor verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf das erste Element eines Double Array, enthält nach Rücksprung die voreingestellten Korrekturfaktoren. Das Array benötigt Platz für 2048 double Werte

Rückgabe

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

5.16.3.17 GOMDBTS2048_substitutionGeoGetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGeoGetIntegralFactor (
            int handle,
            double * factor )
```

Diese Methode liefert den aktuellen Substitutionsfaktor für den integralen Sensor. Dieser Faktor findet bei der aktuellen Messung Verwendung. Falls die Substitutionskorrektur ausgeschaltet ist, wird hier immer der Wert 1.0 zurückgegeben. Wenn der voreingestellte Substitutionsfaktor ermittelt werden soll, der aktuell vorhanden ist (unabhängig davon, ob Substitutionskorrektur aktiv oder nicht), dann sollte die Methode substitutionGeoGetPresetIntegralFactor verwendet werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf Doublewert, enthält den Substitutionsfaktor.

Rückgabe

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

5.16.3.18 GOMDBTS2048_substitutionGeoGetPresetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGeoGetPresetIntegralFactor (
             int handle,
             double * factor )
```

Diese Methode liefert den voreingestellten Substitutionsfaktor für den integralen Sensor, unabhängig davon, ob die Substitutionskorrektur eingeschaltet ist, oder nicht. Dieser Faktor findet nur dann Verwendung, wenn die Substitutionskorrektur aktiviert ist. Für den bei der Messung tatsächlich verwendeten Korrekturfaktor sollte die Methode substitutionGetIntegralFactor verwendet werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*factor*	Pointer auf Doublewert, enthält den Substitutionsfaktor.

Rückgabe

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

5.16.3.19 GOMDBTS2048_substitutionGeoSetComment()

```
int __stdcall GOMDBTS2048_substitutionGeoSetComment (
             int handle,
             char * comment )
```

Diese Methode setzt einen Kommentar, der die aktuelle Substitution näher beschreibt. Sie sollte vor dem Speichern der aktuellen Substitutionskorrektur aufgerufen werden, da dieser Kommentar mit abgespeichert wird.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*comment*	Nullterminierter String, der den Kommentar beinhaltet; maximal zulässige Länge inklusive Terminator: 1024 Bytes

Rückgabe

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

5.16.3.20 GOMDBTS2048_substitutionGeoGetComment()

```
int __stdcall GOMDBTS2048_substitutionGeoGetComment (
             int handle,
             char * comment )
```

Diese Methode liefert den Kommentar, der durch die Methode substitutionGeoSetComment gesetzt wurde.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*comment*	Nullterminierter String, muss mit 1024 Bytes allokiert werden; enthält nach Rücksprung den Kommentar

Rückgabe

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

5.16.3.21 GOMDBTS2048_substitutionGeoGetDateTime()

```
int __stdcall GOMDBTS2048_substitutionGeoGetDateTime (
              int handle,
              int * day,
              int * month,
              int * year,
              int * hh,
              int * mm,
              int * ss )
```

Diese Methode liefert das Datum und die Uhrzeit, zu der die aktuelle Substitutitonskorrektur gespeichert wurde.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*day*	Pointer auf Integer, enthält nach Rücksprung den Tag des Datums
out	*month*	Pointer auf Integer, enthält nach Rücksprung den Monat des Datums
out	*year*	Pointer auf Integer, enthält nach Rücksprung das Jahr des Datums
out	*hh*	Pointer auf Integer, enthält nach Rücksprung die Stunden der Uhrzeit
out	*mm*	Pointer auf Integer, enthält nach Rücksprung die Minuten der Uhrzeit
out	*ss*	Pointer auf Integer, enthält nach Rücksprung die Sekunden der Uhrzeit

Rückgabe

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

5.17 Basis-Kalibriermethoden

Die folgenden Methoden sind für die Durchführung von Kalibrierungen nötig.

Funktionen

- int __stdcall GOMDBTS2048_calibLoadFromDevice (int handle)
- int __stdcall GOMDBTS2048_calibSaveToDevice (int handle, int configNumber)
- int __stdcall GOMDBTS2048_calibSetCalibLampFileName (int handle, char ∗filename)
- int __stdcall GOMDBTS2048_calibGetCalibLampFileName (int handle, char ∗filename)
- int __stdcall GOMDBTS2048_calibMeasureSpectral (int handle, double ∗saturation, double ∗countsDark, double ∗countsSignal, double ∗calFactors)
- int __stdcall GOMDBTS2048_calibMeasureIntegral (int handle, double ∗currentDark, double ∗currentSignal, double ∗calFactor)
- int __stdcall GOMDBTS2048_calibSetIntegrationTimeInUs (int handle, int value)
- int __stdcall GOMDBTS2048_calibGetIntegrationTimeInUs (int handle, int ∗value)
- int __stdcall GOMDBTS2048_calibSetCalibrationName (int handle, char ∗name)
- int __stdcall GOMDBTS2048_calibGetCalibrationName (int handle, char ∗name)
- int __stdcall GOMDBTS2048_calibSetCalibMode (int handle, int value)
- int __stdcall GOMDBTS2048_calibGetCalibMode (int handle, int ∗value)
- int __stdcall GOMDBTS2048_calibSetHighResolutionMode (int handle, int value)
- int __stdcall GOMDBTS2048_calibGetHighResolutionMode (int handle, int ∗value)
- int __stdcall GOMDBTS2048_calibCalculateSpectralCalibrationFactors (int handle, double ∗calFactors)

5.17.1 Ausführliche Beschreibung

Die folgenden Methoden sind für die Durchführung von Kalibrierungen nötig. Diese Methoden sind frei aufrufbar, die Speichermethode der Kalibrierkonfiguration ist jedoch passwortgeschützt, da zum Kalibrieren besondere Voraussetzungen erfüllt sein müssen. Zum Speichern stehen die Konfigurationsnummern 15 – 23 zur freien Verfügung. Die Konfigurationsnummern 0 – 14 sind für Werkskalibrierungen reserviert. Die unterschiedlichen Konfigurationsbereiche sind passwortgeschützt (siehe Methode „setPassword“). Die Berechtigung wird aber erst bei Aufruf der Methode „calibSaveToDevice“ überprüft.
Bevor eine Kalibrierung gespeichert werden kann, müssen alle notwendigen Daten mit den in diesem Kapitel aufgeführten Methoden gesetzt worden sein. Wenn bei einer bestehenden Kalibrierung nur einzelne Daten geändert werden sollen empfiehlt es sich die bereits bestehende Kalibrierung zu laden, die zu ändernden Daten danach neu zu setzen und dann die Kalibrierung entweder unter einer neuen Konfigurationsnummer oder unter der alten Konfigurationsnummer zu speichern.
Für den Anfang empfehlen wir die Basis-Kalibriermethoden zu verwenden. Die Applikation kann dann Stück für Stück um den manuellen Kalibriermethoden ergänzt werden.
Die Basis-Kalibriermethoden werden benötigt, um eine Kalibrierung softwaregestützt durchführen zu können (Messung und Speicherung). Dieses Vorgehen wird bei Rekalibrierungen stets empfohlen. Falls die zu setzenden Kalibrierfaktoren nicht bekannt sind, indem sie gesondert ermittelt worden sind, stehen zwei Messmethoden zur Verfügung, mit denen die Kalibrierung durchgeführt werden kann.
Sämtliche Kalibrierdaten werden zunächst in einem temporären Speicher zwischengespeichert und erst mit Aufruf der Speichermethode in den Speicher des Messgeräts übertragen.
Der einfachste Fall einer Kalibrierung kann folgendermaßen erfolgen:
1. Bestehende Kalibrierung laden (calibLoadFromDevice)
2. Kalibrierstandard setzen (calibSetCalibLampFileName)
3. Substitutionskorrektur durchführen (siehe Substitutionsfunktionen)
4. Kalibrierlampe einschalten und Einbrennzeit abwarten
5. Integrationszeit für die spektrale Kalibriermessung einstellen (calibSetIntegrationTimeInUs)
6. Kalibriermessung spektral durchführen (calibMeasureSpectral)
7. Wenn die Sättigung zu niedrig ist, dann zurück zu Schritt 4

8. Kalibriermessung integral durchführen (calibMeasureIntegral)
9. Eventuell neuen Namen für die Kalibrierung definieren (calibSetCalibrationName)
10. Kalibrierung im Messgerät unter einer Konfigurationsnummer ablegen (calibSaveToDevice). Eine bereits an der Stelle bestehende Konfiguration wird überschrieben

Kalibrierung mit meheren Kalibrierstandards (wichtig für BTS2048-UV Geräte)
Wenn mehr als eine Lampe zur Kalibrierung verwendet werden soll, muss die oben Beschriebene Routine leicht angepasst werden. Dies ist besonders bei BTS2048-UV Geräten von Bedeutung da hier die Kalibrierung mit einer Deuterium- und einer Halogen-Lampe empfohlen wird, um den gesamten WL-Bereich abzudecken.
Bevor der Kalibrierstandard gesetzt wird, muss der Kalibriermodus auf den Wert 2 gesetzt werden (calibSetCalib↩ Mode). Dann müssen die Punkte 2 bis 7 in der obigen Liste für jede Lampe einzeln durchgeführt werden. Wichtig: Die Lampen-Daten(im Kalibrierstandard) sollten dürfen nur für den WL-Bereich definiert sein, in dem diese für die Kalibrierung verwendet werden (Der WL-Bereich für Halogen Lampen sollte 350nm nicht unterschreiten).
Wenn alle Lampen vermessen wurden, muss die Routine calibCalculateSpectralCalibrationFactors() aufgerufen werden. Die einzelnen spektralen Messungen werden dann zusammengefasst und das berechnete Spektrum ausgegeben.
Danach kann in der obigen Liste bei Punkt 8 fortgefahren werden. Beachten Sie, dass der richtige Kalibrierstandard für die integrale Messung gesetzt ist (calibGetLampFileName). Achtung: Die verwendete Lampe sollte den gesamten WL-Bereich der Diode abdecken (bei BTS2048-UV Geräten wird hier eine Deuterium Lampe empfohlen).

5.17.2 C++ Aufrufbeispiel

Kalibrierdaten und Bezeichnungen der Kalibrierungen laden.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                                    //
      initialization
char value[32];
GOMDBTS2048_calibLoadFromDevice(handle);                                //load
       calibration data
GOMDBTS2048_calibGetCalibrationName(handle, value);
//do anything with values
GOMDBTS2048_releaseHandle(handle);                                       //release
       handle
```

5.17.3 Dokumentation der Funktionen

5.17.3.1 GOMDBTS2048_calibLoadFromDevice()

```
int __stdcall GOMDBTS2048_calibLoadFromDevice (
             int handle )
```

Diese Methode lädt die Kalibrierdaten der aktuell ausgewählten Kalibrierkonfiguration („setCalibrationEntry↩ Number") aus dem Messgerät in den Zwischenspeicher. Die Daten können danach mit den Get-Methoden aus dem Zwischenspeicher ausgelesen werden. Bei Änderung von einzelnen Werten einer Kalibrierkonfiguration empfiehlt es sich, die bestehende Konfiguration zuerst zu laden, Änderungen durchzuführen und danach wieder zu speichern. Das Laden der Kalibrierdaten kann mehrere Sekunden bis 1 Minute dauern.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.17.3.2 GOMDBTS2048_calibSaveToDevice()

```
int __stdcall GOMDBTS2048_calibSaveToDevice (
            int handle,
            int configNumber )
```

Die aktuell definierten Kalibrierdaten werden im Messgerät auf der definierten Position abgelegt. Vor dem Speichern wird geprüft, ob alle notwendigen Parameter gesetzt worden sind. Andernfalls wird ein Fehlercode zurückgegeben. Der „userspezifische“ Speicher ist definiert von Nummer 15 – 23. Die Positionen 0 – 14 sind für Werkskalibrierungen der Firma Gigahertz-Optik GmbH reserviert.
Die Verwendung dieser Methode wird zurückgewiesen wenn das anfangs gesetzte Freigabepasswort nicht für Kalibrierungen zulässig ist. Der Speichervorgang kann bis zu einer Minute dauern.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*configNumber*	Integer Wert; Konfigurationsnummer unter der diese Kalibrierung im Gerät abgelegt werden soll, 0 – 14 reserviert für Werkskalibrierungen, 15 – 23 frei verfügbar

Rückgabe

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

5.17.3.3 GOMDBTS2048_calibSetCalibLampFileName()

```
int __stdcall GOMDBTS2048_calibSetCalibLampFileName (
            int handle,
            char * filename )
```

Lädt Kalibrierlampendaten aus einer externen Datei und verwendet diese Daten zur Kalibreirung. Wird diese Methode verwendet, darf die Methode „calibAzSetCalibLamp“ nicht verwendet werden. Wenn die Kalibrierfaktoren mittels der Kalibriermessmethoden ermittelt werden sollen, muss diese Methode anstelle von „calibAzSetCalibLamp“ verwendet werden.
Die Kalibrierlampendaten müssen in der Einheit [W] vorliegen. Das Format für Kalibrierlampendateien ist wie folgt:
Format: ANSI Textdatei
Zeile: (optional) Kommentarzeile, gekennzeichnet durch „//“ oder „;“ am Zeilenanfang
folgende Zeilen: jeweils in einer eigenen Zeile ein Eintrag (Wellenlänge und zugehöriger Lampenwert durch Tabulator getrennt)
Beispiel:
//Kommentarzeile
250 124,365
255 166,447
260 215,786
265 278,089
...

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*filename*	nullterminierter String; enthält den kompletten Pfad zur Lampendatei

Rückgabe

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

5.17.3.4 GOMDBTS2048_calibGetCalibLampFileName()

```
int __stdcall GOMDBTS2048_calibGetCalibLampFileName (
            int handle,
            char * filename )
```

Liefert den zuvor definierten Kalibrierlampendateinamen mit komplettem Pfad. Wenn die Klaibrierlampendaten mittels der Methode „calibAzSetCalibLamp“ übergeben wurden, dann ist der Dateiname leer.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*filename*	nullterminierter String; enthält den kompletten Pfad zur Lampendatei, der String muss mit 2048 Bytes allokiert werden.

Rückgabe

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

5.17.3.5 GOMDBTS2048_calibMeasureSpectral()

```
int __stdcall GOMDBTS2048_calibMeasureSpectral (
            int handle,
            double * saturation,
            double * countsDark,
            double * countsSignal,
            double * calFactors )
```

Mit dieser Methode können die spektralen Kalibrierfaktoren durch eine Kalibriermessung ermittelt werden. Zuvor muss die Kalibrierlampe mittels „calibSetCalibLampFileName“ gesetzt worden sein.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*saturation*	Pointer auf double Wert; enthält die Aussteuerung der spektralen Einheit bei der Kalibriermessung

Parameter

out	*countsDark*	Pointer auf erstes Element eines Double Array, enhält das Dunkelsignal der spektralen Einheit bei der Kalibriermessung; es muss Speicher für 2048 double Werte bereigestellt werden
out	*countsSignal*	Pointer auf erstes Element eines Double Array, enhält das Nutzsignal der spektralen Einheit bei der Kalibriermessung; es muss Speicher für 2048 double Werte bereigestellt werden
out	*calFactors*	Pointer auf erstes Element eines Double Array, enhält die ermittelten Kalibrierfaktoren der spektralen Einheit bei der Kalibriermessung; es muss Speicher für 2048 double Werte bereigestellt werden

Rückgabe

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

5.17.3.6 GOMDBTS2048_calibMeasureIntegral()

```
int __stdcall GOMDBTS2048_calibMeasureIntegral (
            int handle,
            double * currentDark,
            double * currentSignal,
            double * calFactor )
```

Mit dieser Methode kann der spektrale Kalibrierfaktor durch eine Kalibriermessung ermittelt werden. Zuvor muss die Kalibrierlampe mittels „calibSetCalibLampFileName“ gesetzt worden sein.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*currentDark*	Pointer auf Double Wert, enhält den Dunkelstrom der integralen Einheit bei der Kalibriermessung
out	*currentSignal*	Pointer auf Double Wert, enhält den Nutzstrom der integralen Einheit bei der Kalibriermessung
out	*calFactor*	Pointer auf Double Wert, enhält den ermittelten Kalibrierfaktor der integralen Einheit bei der Kalibriermessung

Rückgabe

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

5.17.3.7 GOMDBTS2048_calibSetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_calibSetIntegrationTimeInUs (
            int handle,
            int value )
```

Mit dieser Methode wird die bei der spektralen Kalibriermessung zu verwendende Integrationszeit definiert. Wenn die Sättigung zu niedrig ist (sollte mindestens 54% betragen), sollte die Integrationszeit größer gewählt werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	double Wert, enthält die Integrationszeit in Microsekunden

Rückgabe

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

5.17.3.8 GOMDBTS2048_calibGetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_calibGetIntegrationTimeInUs (
             int handle,
             int * value )
```

Mit dieser Methode wird die bei der spektralen Kalibriermessung zu verwendende Integrationszeit definiert. Wenn die Sättigung zu niedrig ist (sollte mindestens 54% betragen), sollte die Integrationszeit größer gewählt werden.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf double Wert, enthält die Integrationszeit in Microsekunden

Rückgabe

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

5.17.3.9 GOMDBTS2048_calibSetCalibrationName()

```
int __stdcall GOMDBTS2048_calibSetCalibrationName (
             int handle,
             char * name )
```

Mit dieser Methode wird der Name der Kalibrierkonfiguration definiert, der auch im Konfigurationsfenster angezeigt wird. Der Name sollte eindeutig sein um nicht zu Verwechslungen zu führen. Werkskalibrierungen werden immer nach folgendem Schema benannt: „NAME (EINHEIT)“, wobei NAME einem beliebigen Namen entspricht und EI↩ NHEIT der tatsächliche Einheit der integralen Messeinheit entspricht.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*name*	Nullterminierter String, maximale Länge 31 Zeichen plus Nullterminator

Rückgabe

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

5.17.3.10 GOMDBTS2048_calibGetCalibrationName()

```
int __stdcall GOMDBTS2048_calibGetCalibrationName (
            int handle,
            char * name )
```

Diese Methode liefert die zuvor gesetzte Bezeichnung der Kalibrierung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*name*	nullterminierter String; enthält nach Rücksprung die definierte Bezeichnung der Kalibrierung, Mindestgröße: 32 Bytes

Rückgabe

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

Aufrufbeispiel:

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);
char value[32];
GOMDBTS2048_calibLoadFromDevice(handle);
GOMDBTS2048_calibGetCalibrationName(handle, value);
//do anything with values
GOMDBTS2048_releaseHandle(handle);
```

5.17.3.11 GOMDBTS2048_calibSetCalibMode()

```
int __stdcall GOMDBTS2048_calibSetCalibMode (
            int handle,
            int value )
```

Diese Methode definiert den Kalibriermodus. Der default Wert ist 0 und entspricht der Standard Kalibriermethode.
Mode 0: Standard-Kalibriermethode (default)
Mode 1: Resakalierung. Anstatt alle Kalibriereinträge neu zu setzten, werden hier alle spektralen Einträge lediglich mit einem Faktor so skaliert, dass der gemessene Radiometrische Wert mit dem des Lampenfiles übereinstimmt.
Mode 2: Kalibrierung mit meheren Kalibrierlampen. Bei diesem Mosud können meherer Lampen mit unterschiedlichem WL-Bereich verwendet werden. Für die genauere Vorgehensweise lesen Sie die den Punkt „Kalibrierung mit meheren Kalibrierlampen“ in der Einleitung zu diesem Kapitel.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Parameter

in	*value*	Integer Wert: • 0: Normaler Kalibrierungsmodus • 1: Reskalierung statt Rekalibrierung • 2: Kalibrierung mit meheren Lampen

Rückgabe

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

5.17.3.12 GOMDBTS2048_calibGetCalibMode()

```
int __stdcall GOMDBTS2048_calibGetCalibMode (
            int handle,
            int * value )
```

Diese Methode gibt den Kalibriermodus zurück.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer Wert, enthält nach Rücksprung den Kalibrierungmodus. • 0: Normaler Kalibrierungsmodus • 1: Reskalierung statt Rekalibrierung • 2: Kalibrierung mit meheren Lampen

Rückgabe

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

5.17.3.13 GOMDBTS2048_calibSetHighResolutionMode()

```
int __stdcall GOMDBTS2048_calibSetHighResolutionMode (
            int handle,
            int value )
```

Diese Methode schaltet den hochauflösenden Messmodus für die Funktion calibMeasureSpectral() an und aus. Dabei wird das Spektrum aus mehereren Messungen mit unterschiedlichen Integrationszeiten zusammengesetzt. Der Übergabeparameter gibt dabei den dynamik-Bereich(Anzahl Messungen). Bei 0 wird der Modus deaktiviert.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert: • 0: Hochauflösende Messung wird deaktiviert • Sonst(>0): Angabe des Dynamik-Bereichs (Anzahl Messungen)

Rückgabe

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

5.17.3.14 GOMDBTS2048_calibGetHighResolutionMode()

```
int __stdcall GOMDBTS2048_calibGetHighResolutionMode (
            int handle,
            int * value )
```

Diese Methode gibt den Status der hochauflösenden Kalibriermessung zurück.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Integer Wert: • 0: Hochauflösende Messung daktiviert • Sonst(>0): Dynamik-Bereich(Anzahl Messungen mit unterschiedlicher Integrationszeit)

Rückgabe

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

5.17.3.15 GOMDBTS2048_calibCalculateSpectralCalibrationFactors()

```
int __stdcall GOMDBTS2048_calibCalculateSpectralCalibrationFactors (
            int handle,
            double * calFactors )
```

Diese findet nur im Kalibriermodus 2 Verwendung. Nachdem alle Kalibrierlampen vermessen wurden, muss diese Funktion einmalig aufgerufen werden, um alle spektralen Messungen zu kombinieren und die Kalibrierfaktoren zu berechnen.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*calFactors*	Pointer auf erstes Element eines Double Array, enhält die ermittelten Kalibrierfaktoren der spektralen Einheit bei der Kalibriermessung; es muss Speicher für 2048 double Werte bereigestellt werden

Rückgabe

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

5.18 Manuelle Kalibriermethoden

Die manuellen Kalibriermethoden bieten zusätzliche Funktionalität die über die Basis-Kalibriermethoden hinausgehen.

Funktionen

- int __stdcall GOMDBTS2048_calibNew (int handle)
- int __stdcall GOMDBTS2048_calibTristimulusGetXYZ (int handle, double ∗valuesX, double ∗valuesY, double ∗valuesZ)
- int __stdcall GOMDBTS2048_calibAzSetCalibLamp (int handle, double ∗values, char ∗name)
- int __stdcall GOMDBTS2048_calibAzGetCalibLamp (int handle, double ∗values, char ∗name)
- int __stdcall GOMDBTS2048_calibAzSetTransmissionFileActual (int handle, char ∗absoluteFileName)
- int __stdcall GOMDBTS2048_calibAzSetWeightingFunctionActual (int handle, double ∗values)
- int __stdcall GOMDBTS2048_calibAzGetWeightingFunctionActual (int handle, double ∗values)
- int __stdcall GOMDBTS2048_calibSetCalibrationFactorsSpectral (int handle, double ∗values)
- int __stdcall GOMDBTS2048_calibGetCalibrationFactorsSpectral (int handle, double ∗values)
- int __stdcall GOMDBTS2048_calibSetUnitSpectral (int handle, int value)
- int __stdcall GOMDBTS2048_calibGetUnitSpectral (int handle, int ∗value)
- int __stdcall GOMDBTS2048_calibSetCalibrationFactorIntegral (int handle, double value)
- int __stdcall GOMDBTS2048_calibGetCalibrationFactorIntegral (int handle, double ∗value)
- int __stdcall GOMDBTS2048_calibSetUnitIntegral (int handle, int value)
- int __stdcall GOMDBTS2048_calibGetUnitIntegral (int handle, int ∗value)
- int __stdcall GOMDBTS2048_calibSetFilterAssignment (int handle, int value)
- int __stdcall GOMDBTS2048_calibGetFilterAssignment (int handle, int ∗value)
- int __stdcall GOMDBTS2048_calibSetExternalSphere (int handle, bool value)
- int __stdcall GOMDBTS2048_calibGetExternalSphere (int handle, bool ∗value)
- int __stdcall GOMDBTS2048_calibTristimulusSetXYZ (int handle, double ∗valuesX, double ∗valuesY, double ∗valuesZ)

5.18.1 Ausführliche Beschreibung

Die manuellen Kalibriermethoden bieten zusätzliche Funktionalität die über die Basis-Kalibriermethoden hinausgehen. Alle kalibrierrelevanten Parameter können hiermit manuell ergänzt oder auch geändert werden. Die beiden Methoden können auch kombiniert werden. Es kann Beispielsweise eine bestehende Kalibrierung geladen, die Kalibrierfaktoren über Basiskalibrierumethoden ermittelt und weitere Parameter mit den manuellen Kalibriermethoden gesetzt werden.
Hier ein Beispiel für eine komplett manuelle Kalibrierung. Die spektralen und integralen Kalibrierfaktoren müssen bereits bekannt sein:
1. Eine neue Kalibrierung erstellen und in den Speicher laden (calibNew). Wenn eine neue Kalibrierung erstellt wird muss beachtet werden, dass alle Kalibrierparameter gesetzt werden die mit erforderlich gekennzeichnet sind.
2. Filterposition für die neue auswählen(calibSetFilterAssignment). (erforderlich)
3. Kalibrierlampe für A∗-Korrektur setzen (calibAzSetCalibLamp). (erforderlich)
4. Gewichtungsfunktion der Diode setzen (calibAzSetWeightingFunctionActual) oder bei einer Kalibrierung an der Kugel die Transmissionsdaten der Kugel setzten (calibAzSetTransmissionFileActual). Wird hier nichts gesetzt, wird die Gewichtungsfunktion aus dem Gerät verwendet.
5. Spektrale Kalibrierwerte übertragen (calibSetCalibrationFactorsSpectral) und die spektrale Einheit setzen (calib↩SetUnitSpectral). (erforderlich)
6. Integralen Kalibrierwert übertragen (calibSetCalibrationFactorIntegral) und die integrale Einheit setzen (calib↩SetUnitIntegral). (erforderlich)
7. Wird eine Kugelkalibrieung durchgeführt, muss der Parameter external Sphere auf „true“ gesetzt werden (calib↩SetExternalSphere).
8. Kalibrierung im Messgerät unter einer Konfigurationsnummer ablegen (calibSaveToDevice). Eine bereits an der Stelle bestehende Konfiguration wird überschrieben

5.18.2 C++ Aufrufbeispiel

Kalibrierungen und spektrale Kalibrierfaktoren laden.

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);                                    //
      initialization
double values[2048];
GOMDBTS2048_calibLoadFromDevice(handle);                                //load
       calibration data
GOMDBTS2048_calibGetCalibrationFactorsSpectral(handle, values
      );           //get calibration factors
//do anything with values
GOMDBTS2048_releaseHandle(handle);                                       //release
       handle
```

5.18.3 Dokumentation der Funktionen

5.18.3.1 GOMDBTS2048_calibNew()

```
int __stdcall GOMDBTS2048_calibNew (
             int handle )
```

Löscht den aktuellen Kalibrierzwischenspeicher bzw. legt den Kalibrierzwischenspeicher neu an. Diese Methode muss nicht zwangsweise aufgerufen werden, da der Kalibrierzwischenspeicher mit den „Set“-Methoden bei Nichtexistenz automatisch angelegt wird. Man kann damit lediglich sicherstellen, dass der Zwischenspeicher von bereits im Speicher stehenden alten Kalibrierwerten befreit wird.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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

5.18.3.2 GOMDBTS2048_calibTristimulusGetXYZ()

```
int __stdcall GOMDBTS2048_calibTristimulusGetXYZ (
             int handle,
             double * valuesX,
             double * valuesY,
             double * valuesZ )
```

Diese Methode liefert die mit der Methode „calibTristimulusGetXYZ“ bzw. geladenenen X-, Y-, Z- Tristimulus-Werte.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*valuesX*	Pointer auf das erste Element eines double array der Größe 500, enthält nach Rücksprung bereits definierten Tristumuluswerte für X von 350nm bis 849nm in 1nm Schrittweite -$>$ genau 500 Werte

Parameter

out	*valuesY*	Pointer auf das erste Element eines double array der Größe 500, enthält nach Rücksprung bereits definierten Tristumuluswerte für Y von 350nm bis 849nm in 1nm Schrittweite -> genau 500 Werte
out	*valuesZ*	Pointer auf das erste Element eines double array der Größe 500, enthält nach Rücksprung bereits definierten Tristumuluswerte für Z von 350nm bis 849nm in 1nm Schrittweite -> genau 500 Werte

Rückgabe

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

5.18.3.3 GOMDBTS2048_calibAzSetCalibLamp()

```
int __stdcall GOMDBTS2048_calibAzSetCalibLamp (
             int handle,
             double * values,
             char * name )
```

Mit dieser Methode wird das Kalibrierlampenspektrum abgelegt. Die relativen spektralen Daten sind ausreichend. Für die Kalibrierlampe kann ein Name vergeben werden. Das Kalibrierlampenspektrum wird für die a(z)-Korrektur benötigt, ist für jede Kalibrierung individuell im Messgerät abgelegt und muss daher für jede Kalibrierkonfiguration gesetzt werden. Das Spekturm muss vorliegen in einem Bereich von 350nm bis 849nm mit einer Schrittweite von 1nm. Es werden also genau 500 Werte benötigt. Wenn die Kalibrierlampendaten mittels „calibSetCalibLampFile↩ Name“ übergeben werden, oder wenn die Kalibrierfaktoren mittels einer Kalibriermessung (calibMeasureSpectral und calibMeasureIntegral) ermittelt werden sollen, müssen die Kalibrierlampendaten mit der Methode „calibSet↩ CalibLampFileName“ übergeben werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*values*	double array, enthält das Kalibrierlampenspektrum von 350nm bis 849nm in 1nm Schrittweite -> genau 500 Werte
in	*name*	Nullterminierter String, Name der Kalibrierlampe (maximal 31 Zeichen plus Nullterminator).

Rückgabe

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

5.18.3.4 GOMDBTS2048_calibAzGetCalibLamp()

```
int __stdcall GOMDBTS2048_calibAzGetCalibLamp (
             int handle,
             double * values,
             char * name )
```

Diese Methode liefert das definierte Kalibrierlampenspektrum und den definierten Namen der Kalibrierlampe.

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*values*	Pointer auf das erste Element eines double array der Größe 500, enthält nach Rücksprung das definierten Kalibrierlampenspektrum von 350nm bis 849nm in 1nm Schrittweite -> genau 500 Werte
out	*name*	Nullterminierter String; enthält nach Rücksprung den Namen der Kalibrierlampe, Mindestgröße: 32 Bytes.

Rückgabe

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

5.18.3.5 GOMDBTS2048_calibAzSetTransmissionFileActual()

```
int __stdcall GOMDBTS2048_calibAzSetTransmissionFileActual (
            int handle,
            char * absoluteFileName )
```

Mit dieser Methode kann die Transmissionskurve einer Ulbricht'schen Kugel gesetzt werden. Diese befindet sich in einer Textdatei, zu welcher der voll qualifizierte Filename angegeben werden muss. Das Format für Transmissionsdateien ist wie folgt:
Format: ANSI Textdatei
1. Zeile: (optional) Kommentarzeile, gekennzeichnet durch „//“ oder „;“ am Zeilenanfang
folgende Zeilen: jeweils in einer eigenen Zeile ein Eintrag (Wellenlänge und zugehöriger Transmissionswert durch Tabulator getrennt)
Beispiel:
//Kommentarzeile
250 124,365
255 166,447
260 215,786
265 278,089
...

Parameter

in	*handle*	Integer Wert > 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*absoluteFileName*	Nullterminierter String, enthält den absoluten Filenamen der Transmissionsdatendatei.

Rückgabe

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

5.18.3.6 GOMDBTS2048_calibAzSetWeightingFunctionActual()

```
int __stdcall GOMDBTS2048_calibAzSetWeightingFunctionActual (
            int handle,
            double * values )
```

Diese Methode setzt die Bewertungsfunktions des integralen Sensors. Wird diese Funktion nicht aufgerufen, wird die Gewichtungsfunktion aus dem Gerät verwendet.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*values*	Pointer auf das erste Element eines double array der Größe 500, welches die integrale Gewichtungsfunktion enthält.

Rückgabe

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

5.18.3.7 GOMDBTS2048_calibAzGetWeightingFunctionActual()

```
int __stdcall GOMDBTS2048_calibAzGetWeightingFunctionActual (
            int handle,
            double * values )
```

Diese Methode liefert die zuvor definierte Bewertungsfunktions des integralen Sensors

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*values*	Pointer auf das erste Element eines double array der Größe 500, enthält nach Rücksprung die bereits definierte Bewertungsfunktion des integralen Sensors inkl. Beaufschlagung eventueller Transmissionskurven von 350nm bis 849nm in 1nm Schrittweite -$>$ genau 500 Werte.

Rückgabe

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

5.18.3.8 GOMDBTS2048_calibSetCalibrationFactorsSpectral()

```
int __stdcall GOMDBTS2048_calibSetCalibrationFactorsSpectral (
            int handle,
            double * values )
```

Mit dieser Methode werden die Kalibrierfaktoren der spektralen Einheit definiert. Die spektrale Einheit hat 2048 Pixel. Es muss also ein double array mit 2048 Kalibrierfaktoren übergeben werden (pro Pixel ein Kalibrierfaktor). Die Faktoren werden pro Pixel benötigt in: Absolutwert/Nutzcounts$*$Integrationszeit/Substitutionsfaktor.
Einheiten der Größen:

- Absolutwert siehe Tabelle:

0: W
1: W/m2
2: W/sr
3: W/m2/sr
4: lm
5: lx
6: cd
7: cd/m2
8: MED/h
9: mol/m2/s
10: A
11: cd*sr
12: lm/sr
13: lm/m2
14: pc
15: fc
16: E/s/m2
17: W/cm2
18: W/cm2*sr
19: lm/cm2
20: cd*sr/m2
21: fL
22: sb
23: L
24: nit

- Nutzcounts in cts
- Integrationszeit in Sekunden
- Substitutionsfaktor einheitenlos

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*values*	Double array, enthält die Kalibrierfaktoren für die spektrale Messeinheit, genau 2048 Werte, pro Pixel ein Kalibrierfaktor.

Rückgabe

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

5.18.3.9 GOMDBTS2048_calibGetCalibrationFactorsSpectral()

```
int __stdcall GOMDBTS2048_calibGetCalibrationFactorsSpectral (
              int handle,
              double * values )
```

Diese Methode liefert die zuvor definierten Kalibrierfaktoren der spektralen Einheit.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*values*	Pointer auf das erste Element eines double array der Größe 2048, enthält nach Rücksprung die bereits definierten Kalibrierfaktoren für alle Pixel -> genau 2048 Werte

Rückgabe

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

5.18.3.10 GOMDBTS2048_calibSetUnitSpectral()

```
int __stdcall GOMDBTS2048_calibSetUnitSpectral (
            int handle,
            int value )
```

Mit dieser Methode wird die Einheit definiert, in der die Kalibrierlampe für die spektrale Messung vorliegt, z.B. W = 0. Es gilt folgende SI-Einheiten-Tabelle:

0: W
1: W/m2
2: W/sr
3: W/m2/sr
4: lm
5: lx
6: cd
7: cd/m2
8: MED/h
9: mol/m2/s
10: A
11: cd*sr
12: lm/sr
13: lm/m2
14: pc
15: fc
16: E/s/m2
17: W/cm2
18: W/cm2*sr
19: lm/cm2
20: cd*sr/m2
21: fL
22: sb
23: L
24: nit

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert, enthält die Einheitsnummer für die spektrale Messeinheit.

Rückgabe

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

5.18.3.11 GOMDBTS2048_calibGetUnitSpectral()

```
int __stdcall GOMDBTS2048_calibGetUnitSpectral (
```

```
            int handle,
            int * value )
```

Diese Methode liefert die zuvor definierte Einheit der spektralen Messeinheit laut Einheitentabelle.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf integer Wert, enthält nach Rücksprung die bereits definierte Einheitsnummer laut Einheitentabelle.

Rückgabe

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

5.18.3.12 GOMDBTS2048_calibSetCalibrationFactorIntegral()

```
int __stdcall GOMDBTS2048_calibSetCalibrationFactorIntegral (
            int handle,
            double value )
```

Mit dieser Methode wird der Kalibrierfaktor der integralen Einheit definiert. Dieser wird benötigt in: Absolutwert/↩ Strommesswert/Substitutionsfaktor. Einheiten der Größen:

- Absolutwert siehe Tabelle:

  0: W
  1: W/m2
  2: W/sr
  3: W/m2/sr
  4: lm
  5: lx
  6: cd
  7: cd/m2
  8: MED/h
  9: mol/m2/s
  10: A
  11: cd*sr
  12: lm/sr
  13: lm/m2
  14: pc
  15: fc
  16: E/s/m2
  17: W/cm2
  18: W/cm2*sr
  19: lm/cm2
  20: cd*sr/m2
  21: fL
  22: sb
  23: L
  24: nit

- Strommesswert in A
- Substitutionsfaktor einheitenlos

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Double Wert, enthält den Kalibrierfaktor für die integrale Messeinheit.

Rückgabe

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

5.18.3.13 GOMDBTS2048_calibGetCalibrationFactorIntegral()

```
int __stdcall GOMDBTS2048_calibGetCalibrationFactorIntegral (
             int handle,
             double * value )
```

Diese Methode liefert den zuvor definierten Kalibrierfaktor der integralen Einheit.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf double Wert, enthält nach Rücksprung den bereits definierten Kalibrierfaktor für die integrale Einheit.

Rückgabe

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

5.18.3.14 GOMDBTS2048_calibSetUnitIntegral()

```
int __stdcall GOMDBTS2048_calibSetUnitIntegral (
             int handle,
             int value )
```

Mit dieser Methode wird die SI-Einheit definiert, auf die der integrale Detektor kalibriert wurde, z.B. lm = 4.
Es gilt folgende SI-Einheiten-Tabelle:

0: W
1: W/m2
2: W/sr
3: W/m2/sr
4: lm
5: lx
6: cd
7: cd/m2
8: MED/h
9: mol/m2/s
10: A
11: cd*sr
12: lm/sr
13: lm/m2
14: pc
15: fc
16: E/s/m2
17: W/cm2
18: W/cm2*sr
19: lm/cm2
20: cd*sr/m2
21: fL
22: sb
23: L
24: nit

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert, enthält die Einheitsnummer für die integrale Messeinheit.

Rückgabe

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

5.18.3.15 GOMDBTS2048_calibGetUnitIntegral()

```
int __stdcall GOMDBTS2048_calibGetUnitIntegral (
            int handle,
            int * value )
```

Diese Methode liefert die zuvor definierte SI-Einheit für die integrale Messeinheit laut Einheitentabelle.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Parameter

`out`	*value*	Pointer auf integer Wert, enthält nach Rücksprung die bereits definierte Einheitsnummer laut Einheitentabelle: 0: W 1: W/m2 2: W/sr 3: W/m2/sr 4: lm 5: lx 6: cd 7: cd/m2 8: MED/h 9: mol/m2/s 10: A 11: cd*sr 12: lm/sr 13: lm/m2 14: pc 15: fc 16: E/s/m2 17: W/cm2 18: W/cm2*sr 19: lm/cm2 20: cd*sr/m2 21: fL 22: sb 23: L 24: nit

Rückgabe

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

5.18.3.16 GOMDBTS2048_calibSetFilterAssignment()

```
int __stdcall GOMDBTS2048_calibSetFilterAssignment (
              int handle,
              int value )
```

Beschreibung: Das BTS2048 hat ein Filterrad mit 4 bzw. 8 (je nach Ausstattung) Filterpositionen (Offen, OD1, OD2, Geschlossen). Eine Kalibrierung benötigt immer eine zugeordnete Filterposition, für die diese Kalibrierung gültig ist. Diese Filterradposition wird immer zur Messung verwendet, wenn nicht der Filter nach Auswahl eines Kalibriereintrags mit „setFilterPosition“ neu gesetzt wird. Die Filterradpositionszuordnung kann mit dieser Methode definiert werden. Im folgenden sind die Filterradpositionen mit ihren zugehörigen Filtern aufgelistet:

- 0: Geschlossen
- 1: OD2
- 2: OD1
- 3: Offen

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*value*	Integer Wert, enthält die Filterradposition, mögliche Werte 0 - 3.

Rückgabe

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

5.18.3.17 GOMDBTS2048_calibGetFilterAssignment()

```
int __stdcall GOMDBTS2048_calibGetFilterAssignment (
            int handle,
            int * value )
```

Diese Methode liefert die zur Kalibrierung zugeordnete Filterradposition.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf integer Wert, enthält nach Rücksprung die zugeordnete Filterradposition: • 0: Geschlossen • 1: OD2 • 2: OD1 • 3: Offen

Rückgabe

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

5.18.3.18 GOMDBTS2048_calibSetExternalSphere()

```
int __stdcall GOMDBTS2048_calibSetExternalSphere (
            int handle,
            bool value )
```

Diese Methode definiert, ob diese Kalibrierung für den Messaufbau mit einer externen Ulbricht-Kugel Gültigkeit hat. Wenn dieser Parameter nicht explizit gesetzt wird oder bereits gesetzt wurde, dann wird hier als default „false“ (keine Kugel) angenommen.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Parameter

in	*value*	Boolean Wert: • true: Kalibrierung für Messaufbau mit Ulbricht-Kugel • false: Messaufbau enthält keine Ulbricht-Kugel

Rückgabe

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

5.18.3.19 GOMDBTS2048_calibGetExternalSphere()

```
int __stdcall GOMDBTS2048_calibGetExternalSphere (
             int handle,
             bool * value )
```

Diese Methode liefert die Info, ob die Kalibrierung für einen Messaufbau mit einer Ulbricht-Kugel definiert wurde.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*value*	Pointer auf Boolean Wert, enthält nach Rücksprung die Info über eine externe Kugel: • true: Messaufbau mit Ulbricht-Kugel, • false: Messaufbau ohne Ulbricht-Kugel

Rückgabe

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

5.18.3.20 GOMDBTS2048_calibTristimulusSetXYZ()

```
int __stdcall GOMDBTS2048_calibTristimulusSetXYZ (
             int handle,
             double * valuesX,
             double * valuesY,
             double * valuesZ )
```

Diese Methode setzt die Tristimulus Bewertungskurven für das Gerät. Diese Funktionen werden verwendet, um die Farbwerte X, Y und Z zu berechnen. Die Kurven decken den Bereich 350nm bis 849nm in 1nm Schritten ab -> genau 500 Werte wird die Gewichtungsfunktion aus dem Gerät verwendet.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Parameter

in	*valuesX*	Pointer auf das erste Element eines double array der Größe 500, welches die Bewertungskurve für die Farbberechnung des X-Wertes enthält
in	*valuesY*	Pointer auf das erste Element eines double array der Größe 500, welches die Bewertungskurve für die Farbberechnung des Y-Wertes enthält
in	*valuesZ*	Pointer auf das erste Element eines double array der Größe 500, welches die Bewertungskurve für die Farbberechnung des Z-Wertes enthält

Rückgabe

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

5.19 Wellenlängen Kalibriermethoden

Funktionen

- int __stdcall GOMDBTS2048_calibSetWavelengthMapping (int handle, double ∗values)
- int __stdcall GOMDBTS2048_calibGetWavelengthMapping (int handle, double ∗values)
- int __stdcall GOMDBTS2048_calibWavelengthMeasureLamp (int handle, int lampnumber)
- int __stdcall GOMDBTS2048_calibWavelengthCalculateMapping (int handle, double ∗mapping)
- int __stdcall GOMDBTS2048_calibWavelengthSaveMapping (int handle)

5.19.1 Ausführliche Beschreibung

5.19.2 Dokumentation der Funktionen

5.19.2.1 GOMDBTS2048_calibSetWavelengthMapping()

```
int __stdcall GOMDBTS2048_calibSetWavelengthMapping (
            int handle,
            double * values )
```

Mit dieser Methode kann die Pixel-Wellenlängenzuordnung verändert werden. Achtung: Die Wellenlängenzuordnung wird mit dieser Methode temporär in der DLL abgelegt. Um sie im Gerät zu speichern, muss die Methode calibWavelengthSaveMapping aufgerufen werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*values*	Double array, enthält die Pixel zu Wellenlängen Zuordnung für alle 2048 Pixel des BTS2048 in aufsteigender Reihenfolge. Der erste Wert des Array entspricht dem ersten Pixel, der letzte (2048.) Wert des Array entspricht dem letzten Pixel.

Rückgabe

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

5.19.2.2 GOMDBTS2048_calibGetWavelengthMapping()

```
int __stdcall GOMDBTS2048_calibGetWavelengthMapping (
            int handle,
            double * values )
```

Diese Methode liefert die aktuelle Pixel-Wellenlängen-Zuordnung.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*values*	Pointer auf das erste Element eines double array der Größe 2048, enthält nach Rücksprung die Pixel-Wellenlängen-Zuordnung.

Rückgabe

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

5.19.2.3 GOMDBTS2048_calibWavelengthMeasureLamp()

```
int __stdcall GOMDBTS2048_calibWavelengthMeasureLamp (
             int handle,
             int lampnumber )
```

Mit dieser Methode wird eine bestimme Kalibrierlampe vermessen. Integrationszeit, Übersteuerung und andere Faktoren, werden automatisch bestimmt. Für eine erfolgreiche Wellenlängenkalibrierung muss jeder Lampe einmal vermessen werden. Die Zuordung der Lampennummer ist wie folgt:

- Lampennummer 0: HgAr-VL-Lampe
- Lampennummer 1: Ne-VL-Lampe
- Lampennummer 2: Kr-VL-Lampe

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
in	*lampnumber*	integer Lampennummer, siehe Beschreibung.

Rückgabe

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

5.19.2.4 GOMDBTS2048_calibWavelengthCalculateMapping()

```
int __stdcall GOMDBTS2048_calibWavelengthCalculateMapping (
             int handle,
             double * mapping )
```

Wenn alle Lampen vermessen wurden, kann mit dieser Methode die Wellenlängen-Pixel Zuordnung berechnet werden. Stellen Sie sicher, dass die Datei „calibWavelengthBTS2048.gdf“ von der DLL gefunden wird. Dazu muss die Datei entweder im selber Ordner liegen oder unter „Dokumente\ Gigahertz-Optik\ datacalib“. Die Berechnug und Zuordnung erfolgt dann automatisch. Diese wird temporär in der DLL abgelegt. Um sie im Gerät zu speichern, muss die Methode calibWavelengthSaveMapping aufgerufen werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.
out	*mapping*	Pointer auf das erste Element eines double array der Größe 2048, enthält nach Rücksprung die Pixel-Wellenlängen-Zuordnung

Rückgabe

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

5.19.2.5 GOMDBTS2048_calibWavelengthSaveMapping()

```
int __stdcall GOMDBTS2048_calibWavelengthSaveMapping (
            int handle )
```

Diese Methode speichert die temporär abgelegte Wellenlängenzuordnung im BTS2048. Und überschreibt damit die alte Wellenlängenzuordnung. Beachten Sie, dass diese nicht wiederhergestellt werden kann. Nach Änderung der Wellenlängenzuordnung sind sämtliche im BTS2048 abgespeicherten Kalibrierungen ungültig und müssen erneut durchgeführt werden.

Parameter

in	*handle*	Integer Wert $>$ 0 zur eindeutigen Identifikation des instanziierten BTS2048; dieser Wert wird von der Methode getHandle zurückgeliefert.

Rückgabe

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