Gigahertz-Optik

Programming manual

S-SDK-BTS2048

Contents

1 Information **1**

1.1 Disclaimer 1

1.2 Warranty 1

1.3 License 2

1.4 Overview 2

1.5 Contact information of Gigahertz-Optik 2

1.6 System Requirements 2

1.7 Installation 2

1.8 System preparation 3

2 Change History **4**

3 Errors and Warnings **10**

3.1 Errors 10

3.2 Warnings 11

4 Example - How to import DLL in your application **13**

4.1 C++ 13

4.1.1 BTS2048Example.cpp 13

4.1.2 BTS2048Import.cpp 14

4.1.3 BTS2048Import.h 16

4.2 More Examples 16

5 Module Documentation 17

5.1 Method information . . . 17

5.2 Standard SDK Methods . . . 18

5.2.1 Detailed Description . . . 18

5.2.2 C++ Aufrufbeispiel . . . 18

5.2.3 Function Documentation . . . 19

5.2.3.1 GOMDBTS2048_setPassword() . . . 19

5.2.3.2 GOMDBTS2048_getDLLVersion() . . . 19

5.2.3.3 GOMDBTS2048_getDeviceList() . . . 19

5.2.3.4 GOMDBTS2048_getHandle() . . . 20

5.2.3.5 GOMDBTS2048_releaseHandle() . . . 21

5.2.3.6 GOMDBTS2048_getFirmwareVersion() . . . 21

5.2.3.7 GOMDBTS2048_getSerialNumber() . . . 21

5.2.3.8 GOMDBTS2048_getType() . . . 22

5.2.3.9 GOMDBTS2048_isBTS2048VL() . . . 22

5.2.3.10 GOMDBTS2048_isBTS2048BS() . . . 22

5.2.3.11 GOMDBTS2048_isBTS2048UV() . . . 23

5.2.3.12 GOMDBTS2048_isBTS2048VLTEC() . . . 23

5.2.3.13 GOMDBTS2048_isBTS2048Type() . . . 24

5.2.3.14 GOMDBTS2048_setCooling() . . . 24

5.2.3.15 GOMDBTS2048_getCoolingState() . . . 25

5.2.3.16 GOMDBTS2048_hasCooling() . . . 25

5.2.3.17 GOMDBTS2048_getMaxADC() . . . 26

5.2.3.18 GOMDBTS2048_getNoOfPixels() . . . 26

5.2.3.19 GOMDBTS2048_setIPAddress() . . . 27

5.2.3.20 GOMDBTS2048_getIPAddress() . . . 27

5.2.3.21 GOMDBTS2048_isDHCP() . . . 28

5.2.3.22 GOMDBTS2048_setDHCPServer() . . . 28

5.2.3.23 GOMDBTS2048_isDHCPServer() . . . 28

5.2.3.24 GOMDBTS2048_isConnected() . . . 29

5.3 Common measurement settings 30
5.3.1 Detailed Description 30
5.3.2 Function Documentation 30
5.3.2.1 GOMDBTS2048_saveConfig() 30
5.3.2.2 GOMDBTS2048_loadConfig() 31
5.3.2.3 GOMDBTS2048_saveConfigAsDefault() 31
5.3.2.4 GOMDBTS2048_setCalibrationEntryNumber() 31
5.3.2.5 GOMDBTS2048_getSelectedCalibrationEntryNumber() 32
5.3.2.6 GOMDBTS2048_readCalibrationEntryInfo() 32
5.3.2.7 GOMDBTS2048_getMeasurementQuantity() 33
5.3.2.8 GOMDBTS2048_isMeasurementQuantity() 33
5.3.2.9 GOMDBTS2048_getSelectedMeasurementQuantity() 34
5.3.2.10 GOMDBTS2048_isMultiMeasurement() 34
5.3.2.11 GOMDBTS2048_isOORSLCorrectionMeasurement() 35
5.3.2.12 GOMDBTS2048_isSLMCorrectionMeasurement() 35
5.3.2.13 GOMDBTS2048_setMeasurementMode() 36
5.3.2.14 GOMDBTS2048_getMeasurementMode() 36
5.3.2.15 GOMDBTS2048_setSpectralIntegralSynch() 37
5.3.2.16 GOMDBTS2048_isSpectralIntegralSynch() 37
5.3.2.17 GOMDBTS2048_setDistance() 37
5.3.2.18 GOMDBTS2048_getDistance() 38
5.3.2.19 GOMDBTS2048_getFilterName() 38
5.3.2.20 GOMDBTS2048_getFilterNameforCalibration() 39
5.4 Spectral measurement setting 40
5.4.1 Detailed Description 40
5.4.2 Function Documentation 40
5.4.2.1 GOMDBTS2048_spectralSetEnabled() 40
5.4.2.2 GOMDBTS2048_spectralIsEnabled() 41
5.4.2.3 GOMDBTS2048_spectralSetOffsetMode() 41
5.4.2.4 GOMDBTS2048_spectralGetOffsetMode() 42

5.4.2.5 GOMDBTS2048_OORSLCorrectionSetMode() 42
5.4.2.6 GOMDBTS2048_OORSLCorrectionGetMode() 43
5.4.2.7 GOMDBTS2048_spectralGetIntegrationTimeRangeInus() 43
5.4.2.8 GOMDBTS2048_spectralSetIntegrationTimeInus() 44
5.4.2.9 GOMDBTS2048_spectralGetIntegrationTimeInus() 44
5.4.2.10 GOMDBTS2048_spectralSetMeasurementTimeInUs() 45
5.4.2.11 GOMDBTS2048_spectralGetMeasurementTimeInUs() 45
5.4.2.12 GOMDBTS2048_spectralSetDynamicTimeMode() 45
5.4.2.13 GOMDBTS2048_spectralGetDynamicTimeMode() 46
5.4.2.14 GOMDBTS2048_spectralSetMaxIntegrationTimeInUs() 46
5.4.2.15 GOMDBTS2048_spectralGetMaxIntegrationTimeInUs() 47
5.4.2.16 GOMDBTS2048_spectralSetMaxMeasurementTimeInUs() 47
5.4.2.17 GOMDBTS2048_spectralGetMaxMeasurementTimeInUs() 48
5.4.2.18 GOMDBTS2048_spectralSetNrOfScans() 48
5.4.2.19 GOMDBTS2048_spectralGetNrOfScans() 48
5.4.2.20 GOMDBTS2048_setWavelengthRange() 49
5.4.2.21 GOMDBTS2048_getWavelengthRange() 49
5.4.2.22 GOMDBTS2048_getMinValidWavelength() 50
5.4.2.23 GOMDBTS2048_getMaxValidWavelength() 50
5.5 Spectral correction methods & filter 51
5.5.1 Detailed Description 51
5.5.2 Function Documentation 51
5.5.2.1 GOMDBTS2048_spectralSetScaleWithIntegralMode() 51
5.5.2.2 GOMDBTS2048_spectralGetScaleWithIntegralMode() 52
5.5.2.3 GOMDBTS2048_spectralSetScaleWithVLambda() 52
5.5.2.4 GOMDBTS2048_spectralIsScaleWithVLambda() 53
5.5.2.5 GOMDBTS2048_spectralSetPixelLinearization() 53
5.5.2.6 GOMDBTS2048_spectralIsPixelLinearization() 53
5.5.2.7 GOMDBTS2048_spectralSetBandwidthCorrection() 54
5.5.2.8 GOMDBTS2048_spectralIsBandwidthCorrection() 54

5.5.2.9 GOMDBTS2048_spectralSetSavitzkyGolayFilter() 55
5.5.2.10 GOMDBTS2048_spectralIsSavitzkyGolayFilter() 55
5.5.2.11 GOMDBTS2048_spectralSetNoiseReduction() 56
5.5.2.12 GOMDBTS2048_spectralIsNoiseReduction() 56
5.5.2.13 GOMDBTS2048_spectralSetDarkThreshold() 56
5.5.2.14 GOMDBTS2048_spectralGetDarkThreshold() 57
5.5.2.15 GOMDBTS2048_spectralSetObserver10Degree() 57
5.5.2.16 GOMDBTS2048_spectralIsObserver10Degree() 58
5.5.2.17 GOMDBTS2048_spectralSetAdvancedNoiseReduction() 58
5.5.2.18 GOMDBTS2048_spectralIsAdvancedNoiseReduction() 59
5.5.2.19 GOMDBTS2048_setPreciseCountCalculation() 59
5.5.2.20 GOMDBTS2048_getPreciseCountCalculation() 59
5.6 Integral measurement settings 61
5.6.1 Detailed Description 61
5.6.2 Function Documentation 61
5.6.2.1 GOMDBTS2048_integralSetEnabled() 61
5.6.2.2 GOMDBTS2048_integralIsEnabled() 61
5.6.2.3 GOMDBTS2048_integralGetIntegrationTimeRangeInMs() 62
5.6.2.4 GOMDBTS2048_integralSetIntegrationTimeInUs() 62
5.6.2.5 GOMDBTS2048_integralGetIntegrationTimeInUs() 63
5.6.2.6 GOMDBTS2048_setIntegralRange() 64
5.6.2.7 GOMDBTS2048_getIntegralRange() 64
5.6.2.8 GOMDBTS2048_integralSetRangeWaitTimeInMs() 65
5.6.2.9 GOMDBTS2048_integralGetRangeWaitTimeInMs() 65
5.6.2.10 GOMDBTS2048_integralSetAzMode() 66
5.6.2.11 GOMDBTS2048_integralGetAzMode() 67
5.6.2.12 GOMDBTS2048_integralSetAzSpecific() 67
5.6.2.13 GOMDBTS2048_integralGetAzSpecific() 67
5.7 Methods for color calculation 69
5.7.1 Detailed Description 69

5.7.2 Function Documentation 69
5.7.2.1 GOMDBTS2048_setColorCalculation() 69
5.7.2.2 GOMDBTS2048_isColorCalculation() 69
5.7.2.3 GOMDBTS2048_setColorCalculationMode() 70
5.7.2.4 GOMDBTS2048_getColorCalculationMode() 70
5.7.2.5 GOMDBTS2048_setColorCalculationOptimizationMode() 71
5.7.2.6 GOMDBTS2048_getColorCalculationOptimizationMode() 71
5.7.2.7 GOMDBTS2048_setCTLimitCheckActive() 72
5.7.2.8 GOMDBTS2048_isCTLimitCheckActive() 72
5.7.2.9 GOMDBTS2048_setDeltaUVLimit() 73
5.7.2.10 GOMDBTS2048_getDeltaUVLimit() 73
5.8 Measurement methods 75
5.8.1 Detailed Description 75
5.8.2 Function Documentation 75
5.8.2.1 GOMDBTS2048_measure() 75
5.8.2.2 GOMDBTS2048_spectralEvaluateIntegrationTimeInus() 76
5.8.2.3 GOMDBTS2048_measureGetCountsPixelFast() 76
5.8.2.4 GOMDBTS2048_spectralMeasureOffset() 77
5.8.2.5 GOMDBTS2048_spectralSaveStaticOffset() 77
5.8.2.6 GOMDBTS2048_spectralLoadStaticOffset() 78
5.8.2.7 GOMDBTS2048_spectralMeasureOffsetInDarkPosition() 78
5.8.2.8 GOMDBTS2048_spectralMeasurePremeasuredOffset() 78
5.8.2.9 GOMDBTS2048_spectralExportPremeasuredOffset() 79
5.8.2.10 GOMDBTS2048_spectralImportPremeasuredOffset() 79
5.8.2.11 GOMDBTS2048_spectralDeleteOffset() 79
5.8.2.12 GOMDBTS2048_OORSLCorrectionMeasureFactors() 80
5.8.2.13 GOMDBTS2048_integralMeasureOffset() 80
5.8.2.14 GOMDBTS2048_integralMeasureOffsetInDarkPosition() 81
5.8.2.15 GOMDBTS2048_setFilterPosition() 81
5.8.2.16 GOMDBTS2048_getFilterPosition() 82

5.8.2.17 GOMDBTS2048_integralSeriesMeasure() 82
5.8.2.18 GOMDBTS2048_integralGetSeriesValues() 83
5.8.2.19 GOMDBTS2048_setOut2LowDuringIntegralSeriesMeasurement() 83
5.8.2.20 GOMDBTS2048_getOut2LowDuringIntegralSeriesMeasurement() 84
5.9 Asynchron measurement methods 85
5.9.1 Detailed Description 85
5.9.2 Function Documentation 85
5.9.2.1 GOMDBTS2048_asyncStartMeasurement() 85
5.9.2.2 GOMDBTS2048_asyncStartMeasurementWithTime() 85
5.9.2.3 GOMDBTS2048_asyncGetProgress() 87
5.9.2.4 GOMDBTS2048_asyncStopMeasurement() 87
5.10 Methods for reading integral measurment values 89
5.10.1 Detailed Description 89
5.10.2 Function Documentation 89
5.10.2.1 GOMDBTS2048_integralGetUnit() 89
5.10.2.2 GOMDBTS2048_integralGetSaturation() 89
5.10.2.3 GOMDBTS2048_integralGetLastUsedAz() 91
5.10.2.4 GOMDBTS2048_integralGetValue() 91
5.10.2.5 GOMDBTS2048_integralGetCurrent() 92
5.10.2.6 GOMDBTS2048_integralGetLastUsedRange() 92
5.11 Methods for reading spectral measurment values 93
5.11.1 Detailed Description 93
5.11.2 Function Documentation 93
5.11.2.1 GOMDBTS2048_spectralGetUnit() 93
5.11.2.2 GOMDBTS2048_spectralGetSaturation() 94
5.11.2.3 GOMDBTS2048_getRadiometricValueOverWLRange() 94
5.11.2.4 GOMDBTS2048_getPeak() 94
5.11.2.5 GOMDBTS2048_getFWHM() 95
5.11.2.6 GOMDBTS2048_getCenterWavelength() 95
5.11.2.7 GOMDBTS2048_getCentroidWavelength() 96

5.11.2.8 GOMDBTS2048_spectralGetSpectrumCalibratedWavelength() 96
5.11.2.9 GOMDBTS2048_spectralGetSpectrumCalibratedPixel() 96
5.11.2.10 GOMDBTS2048_spectralGetCountsPixel() 97
5.11.2.11 GOMDBTS2048_spectralGetLambdas() 97
5.11.2.12 GOMDBTS2048_spectralGetSpecmax() 98
5.11.2.13 GOMDBTS2048_spectralGetLastUsedOffset() 98
5.12 Methods for reading common measurment values 100
5.12.1 Detailed Description 100
5.12.2 Function Documentation 100
5.12.2.1 GOMDBTS2048_getTemperature() 100
5.12.2.2 GOMDBTS2048_getLastMaxADC() 100
5.12.2.3 GOMDBTS2048_getLastScaleWithVLFactor() 101
5.13 Methods for reading color measurment values 102
5.13.1 Detailed Description 102
5.13.2 Function Documentation 102
5.13.2.1 GOMDBTS2048_calculateColor() 102
5.13.2.2 GOMDBTS2048_getColor() 102
5.13.2.3 GOMDBTS2048_getDeltaUV() 103
5.13.2.4 GOMDBTS2048_getPurity() 103
5.13.2.5 GOMDBTS2048_getCRI() 104
5.14 Methods for trigger setting 106
5.14.1 Detailed Description 106
5.14.2 Function Documentation 106
5.14.2.1 GOMDBTS2048_setTriggerSource() 106
5.14.2.2 GOMDBTS2048_getTriggerSource() 107
5.14.2.3 GOMDBTS2048_setTriggerInternalLevels() 107
5.14.2.4 GOMDBTS2048_setTriggerMode() 108
5.14.2.5 GOMDBTS2048_getTriggerMode() 108
5.14.2.6 GOMDBTS2048_setTriggerLevel() 108
5.14.2.7 GOMDBTS2048_getTriggerLevel() 109

5.14.2.8 GOMDBTS2048_setTriggerInput() 109
5.14.2.9 GOMDBTS2048_getTriggerInput() 110
5.14.2.10 GOMDBTS2048_isMeasurementFinished() 110
5.14.2.11 GOMDBTS2048_setTriggerTimeoutInMs() 111
5.14.2.12 GOMDBTS2048_getTriggerTimeoutInMs() 111
5.14.2.13 GOMDBTS2048_setOut1LowDuringMeasurement() 112
5.14.2.14 GOMDBTS2048_getOut1LowDuringMeasurement() 112
5.14.2.15 GOMDBTS2048_setTriggerDelay() 113
5.14.2.16 GOMDBTS2048_getTriggerDelay() 113
5.15 Methods for self absorbtion correction DUT (Device Under Test) 114
5.15.1 Detailed Description 114
5.15.2 Function Documentation 114
5.15.2.1 GOMDBTS2048_substitutionEnableCorrection() 114
5.15.2.2 GOMDBTS2048_substitutionIsEnabledCorrection() 115
5.15.2.3 GOMDBTS2048_substitutionMeasurementWithoutTestDevice() 115
5.15.2.4 GOMDBTS2048_substitutionMeasurementWithTestDevice() 116
5.15.2.5 GOMDBTS2048_substitutionSetIntegrationTimeInUs() 117
5.15.2.6 GOMDBTS2048_substitutionGetIntegrationTimeInUs() 117
5.15.2.7 GOMDBTS2048_substitutionSetDynamicTimeMode() 117
5.15.2.8 GOMDBTS2048_substitutionGetDynamicTimeMode() 118
5.15.2.9 GOMDBTS2048_substitutionSetHighResolutionMode() 118
5.15.2.10 GOMDBTS2048_substitutionGetHighResolutionMode() 119
5.15.2.11 GOMDBTS2048_substitutionSaveFactors() 119
5.15.2.12 GOMDBTS2048_substitutionLoadFactors() 119
5.15.2.13 GOMDBTS2048_substitutionGetLoadedFilename() 120
5.15.2.14 GOMDBTS2048_substitutionGetSpectralFactor() 120
5.15.2.15 GOMDBTS2048_substitutionGetSpectralFactors() 121
5.15.2.16 GOMDBTS2048_substitutionGetPresetSpectralFactors() 121
5.15.2.17 GOMDBTS2048_substitutionGetIntegralFactor() 121
5.15.2.18 GOMDBTS2048_substitutionGetPresetIntegralFactor() 122

5.15.2.19 GOMDBTS2048_substitutionSetComment() 122
5.15.2.20 GOMDBTS2048_substitutionGetComment() 123
5.15.2.21 GOMDBTS2048_substitutionGetDateTime() 123
5.16 Methods for self absorbtion correction geometrie 125
5.16.1 Detailed Description 125
5.16.2 Function Documentation 125
5.16.2.1 GOMDBTS2048_substitutionGeoEnableCorrection() 125
5.16.2.2 GOMDBTS2048_substitutionGeoIsEnabledCorrection() 126
5.16.2.3 GOMDBTS2048_substitutionGeoMeasurementWithoutTestDevice() 126
5.16.2.4 GOMDBTS2048_substitutionGeoMeasurementWithTestDevice() 127
5.16.2.5 GOMDBTS2048_substitutionGeoSetIntegrationTimeInUs() 128
5.16.2.6 GOMDBTS2048_substitutionGeoGetIntegrationTimeInUs() 128
5.16.2.7 GOMDBTS2048_substitutionGeoSetDynamicTimeMode() 128
5.16.2.8 GOMDBTS2048_substitutionGeoGetDynamicTimeMode() 129
5.16.2.9 GOMDBTS2048_substitutionGeoSetHighResolutionMode() 129
5.16.2.10 GOMDBTS2048_substitutionGeoGetHighResolutionMode() 130
5.16.2.11 GOMDBTS2048_substitutionGeoSaveFactors() 130
5.16.2.12 GOMDBTS2048_substitutionGeoLoadFactors() 130
5.16.2.13 GOMDBTS2048_substitutionGeoGetLoadedFilename() 131
5.16.2.14 GOMDBTS2048_substitutionGeoGetSpectralFactor() 131
5.16.2.15 GOMDBTS2048_substitutionGeoGetSpectralFactors() 132
5.16.2.16 GOMDBTS2048_substitutionGeoGetPresetSpectralFactors() 132
5.16.2.17 GOMDBTS2048_substitutionGeoGetIntegralFactor() 132
5.16.2.18 GOMDBTS2048_substitutionGeoGetPresetIntegralFactor() 133
5.16.2.19 GOMDBTS2048_substitutionGeoSetComment() 133
5.16.2.20 GOMDBTS2048_substitutionGeoGetComment() 134
5.16.2.21 GOMDBTS2048_substitutionGeoGetDateTime() 134
5.17 Base calibartion methods 136
5.17.1 Detailed Description 136
5.17.2 Function Documentation 136

5.17.2.1 GOMDBTS2048_calibLoadFromDevice() 136
5.17.2.2 GOMDBTS2048_calibSaveToDevice() 136
5.17.2.3 GOMDBTS2048_calibSetCalibLampFileName() 137
5.17.2.4 GOMDBTS2048_calibGetCalibLampFileName() 137
5.17.2.5 GOMDBTS2048_calibMeasureSpectral() 138
5.17.2.6 GOMDBTS2048_calibMeasureIntegral() 138
5.17.2.7 GOMDBTS2048_calibSetIntegrationTimeInUs() 140
5.17.2.8 GOMDBTS2048_calibGetIntegrationTimeInUs() 140
5.17.2.9 GOMDBTS2048_calibSetCalibrationName() 141
5.17.2.10 GOMDBTS2048_calibGetCalibrationName() 141
5.17.2.11 GOMDBTS2048_calibSetCalibMode() 142
5.17.2.12 GOMDBTS2048_calibGetCalibMode() 142
5.17.2.13 GOMDBTS2048_calibSetHighResolutionMode() 143
5.17.2.14 GOMDBTS2048_calibGetHighResolutionMode() 143
5.17.2.15 GOMDBTS2048_calibCalculateSpectralCalibrationFactors() 143
5.18 Manual calibration methods 145
5.18.1 Detailed Description 145
5.18.2 Function Documentation 145
5.18.2.1 GOMDBTS2048_calibNew() 145
5.18.2.2 GOMDBTS2048_calibTristimulusGetXYZ() 146
5.18.2.3 GOMDBTS2048_calibAzSetCalibLamp() 146
5.18.2.4 GOMDBTS2048_calibAzGetCalibLamp() 147
5.18.2.5 GOMDBTS2048_calibAzSetTransmissionFileActual() 147
5.18.2.6 GOMDBTS2048_calibAzSetWeightingFunctionActual() 148
5.18.2.7 GOMDBTS2048_calibAzGetWeightingFunctionActual() 148
5.18.2.8 GOMDBTS2048_calibSetCalibrationFactorsSpectral() 148
5.18.2.9 GOMDBTS2048_calibGetCalibrationFactorsSpectral() 149
5.18.2.10 GOMDBTS2048_calibSetUnitSpectral() 150
5.18.2.11 GOMDBTS2048_calibGetUnitSpectral() 151
5.18.2.12 GOMDBTS2048_calibSetCalibrationFactorIntegral() 151

5.18.2.13 GOMDBTS2048_calibGetCalibrationFactorIntegral() 152
5.18.2.14 GOMDBTS2048_calibSetUnitIntegral() 153
5.18.2.15 GOMDBTS2048_calibGetUnitIntegral() 153
5.18.2.16 GOMDBTS2048_calibSetFilterAssignment() 154
5.18.2.17 GOMDBTS2048_calibGetFilterAssignment() 155
5.18.2.18 GOMDBTS2048_calibSetExternalSphere() 155
5.18.2.19 GOMDBTS2048_calibGetExternalSphere() 156
5.18.2.20 GOMDBTS2048_calibTristimulusSetXYZ() 156
5.19 Wavelength calibration methods 158
5.19.1 Detailed Description 158
5.19.2 Function Documentation 158
5.19.2.1 GOMDBTS2048_calibSetWavelengthMapping() 158
5.19.2.2 GOMDBTS2048_calibGetWavelengthMapping() 158
5.19.2.3 GOMDBTS2048_calibWavelengthMeasureLamp() 159
5.19.2.4 GOMDBTS2048_calibWavelengthCalculateMapping() 159
5.19.2.5 GOMDBTS2048_calibWavelengthSaveMapping() 160

Chapter 1

Information

Please read this documentation and the disclaimer carefully before using the software.
By installing and using the software, you explicitly and fully acknowledge and agree to this.
Gigahertz-Optik GmbH reserves the right to make changes to this manual without prior notice.

1.1 Disclaimer

This software was developed with utmost care and thoroughly tested on different computers. No errors were noted for the approved product versions. However, it cannot be guaranteed that the software will work perfectly on all types of computers. Completely error-free software is not possible with the current technology level.
Gigahertz-Optik GmbH is not liable if the software does not perfectly fulfill your desired purpose or if it is incompatible with other software on your computer. You are therefore solely responsible for the choice, installation and use as well as for the intended results.
With the exception of damages caused deliberately, Gigahertz-Optik GmbH is not liable for any damages caused by the use or inability to use the software. This also exclusively applies for loss of business profits, business interruptions, loss of business information or any other economic losses, even if Gigahertz-Optik GmbH had been previously advised of the possibility of such damage. The enclosed documentation/help of the software is with no claim of accuracy or completeness.

1.2 Warranty

Gigahertz-Optik GmbH guarantees the delivery of all functions listed in the product description. Any available delivery media are free of any material defects.
We have taken all the necessary and possible steps that are required to keep this software free of viruses, spyware, the so-called “back door entrances” or other harmful code. We do not collect any information about you or your data. We will not deliberately limit you from using the functions of this software or access to your data. This agreement supersedes any non-contractual assurances that we may have explained to you. Any modification to this agreement must be confirmed in writing by both parties.

1.3 License

A license for the full version allows you to use the product on only one workstation. Each concurrent use on another workstation requires an additional product license. The distribution of the product and documentation is prohibited. You are authorized to make a copy of this product for your backup purposes. You may pass on your own software that you have developed using this development package together with the required DLLs for this development package to third parties.

1.4 Overview

This development package provides you with all the tools required (no compiler or integrated software development environments) to directly control a BTS2048 series measurement device from Gigahertz-Optik using C/C++. This is primarily with regards to the communication and control libraries for your BTS2048.
In order to use these libraries, you need a programming environment such as Microsoft Visual Studio, Embarcadero C++ builder, etc.

1.5 Contact information of 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 System Requirements

To use the S-SDK BTS2048 you have to consider the following points:

- Minimum disk space – approx. 10MB
- Operation system: MS Windows XP, MS Windows 7 (32bit/64bit), MS Windows 10 (32bit/64bit)
- C/C++ development environment such as MS Visual Studio, Embarcadero C++ Builder, etc. when programming with C/C++
- free USB port

1.7 Installation

Follow the steps below to install the BTS2048-SDK from the product CD:

- Read this documentation before you begin the installation

- Close all other applications before installing
- Insert the CD in your CD drive or unpack the supplied ZIP file.
- Copy the Gigahertz-Optik folder from the CD or ZIP file to a location of your choice. If you have already got other development packages from Gigahertz-Optik installed, it is recommended to use the same installation path in order to avoid possible conflicts.
- Add the folder “install dir/Gigahertz-Optik/runtime” to your system path. "install dir" hereby corresponds to the base path you added in step 4 above. If you have already got other development packages from Gigahertz-Optik installed, step 5 might not be necessary.

1.8 System preparation

Connect the BTS2048 to your computer. The required drivers are standard windows drivers and will be installed automatically.

Chapter 2

Change History

A list of all modifications of the S-SDK-BTS2048 follows:

- **V2014.1**
  Initial version
- **V2014.2**
  Bugfix: communication problems can come up during initialization under USB
- **V2014.3**
  Bugfix: maximum number of scans set from 10 to 100
- **V2014.4**
  Bugfix: minimum step size set from 1nm to 0.25nm
  Bugfix: memory leaks fixed
- **V2014.5**
  Bugfix: NoOfScans could not be specified in the setup
- **V2014.6**
  New: error code -15003 changed to warning 15003
  New: error codes / warnings added or changed in the documentation
  New: “setColorCalculation” added in the documentation
  New: “isColorCalculation” added in the documentation
  New: “saveConfigAsDefault” in the documentation
  New: “spectralGetLambdas” added in the documentation
- **V2014.7**
  New: new error code -15027
  New: “getMeasurementMode” added in the documentation
  New: “getDeltaUV” added in the documentation
  New: “setDistance” added in the documentation
  New: “getDistance” added in the documentation
  New: “getTemperature” added in the documentation
  New: “spectralSetOffsetMode” added in the documentation
  New: “spectralGetOffsetMode” added in the documentation
  New: “spectralMeasureOffset” added in the documentation
  New: “spectralDeleteOffset” added in the documentation
  Bugfix: memory leak fixed

- **V2014.8**
  New: performance improvements
  New: method “integralSetIntegrationTimeInMs”
  New: method “integralGetIntegrationTimeInMs”
  New: method “getDLLVersion”
  New: method “getFirmwareVersion”
  New: method “getSerialNumber”
  New: method “getType”
  New: method “spectralGetIntegrationTimeRangeInus”
  New: method “integralGetRangeIntegrationTimeRangeInMs”
  New: method “getMaxADC”
  New: method “getNoOfPixels”
  Bugfix: memory leak fixed
  Bugfix: when synchronization between the integral and spectral measurement unit was activated, an integral measurement was always triggered even when the integral measurement unit was deactivated.

- **V2014.9**
  New: performance improvements
  New: new method “measureGetCountsPixelFast”
  New: new method “getDeviceList”
  New: new error codes / warning codes

- **V2014.10**
  New: The SDK is password protected. The password has to be set before getting a handle.
  New: method setPassword
  New: method integralGetCurrent
  New: method spectralEvaluateIntegrationTimeInus
  New: methods spectralSetDynamicTimeMode / spectralGetDynamicTimeMode
  New: method getRadiometricValueOverWLRange
  New: calibration methods (see the chapter on: Calibration methods)
  New: new error codes
  New: methods setOut1LowDuringMeasurement / getOut1LowDuringMeasurement

- **V2014.11**
  New: new error codes
  New: performance improvements

- **V2014.12**
  New: performance improvements
  Bugfix: in V2014.11 lieferte der integrale Sensor bei nicht synchroner Messung immer „0”.

- **V2014.13**
  New: new error code for problems with the correction using VL
  Bugfix: in V2014.12 color values always “-1”

- **V2014.14**
  New: performance improvements
  New: new error codes

- **V2014.15**
  New: Method getLastMaxADC
  Bugfix: Errorcode old: -55 -> Errorcode new: -15007
  Bugfix: Errorcode old: -56 -> Errorcode new: -15008

- **V2014.16**
  New: Method spectralSetSavitzkyGolayFilter
  New: Method spectralIsSavitzkyGolayFilter
  New: Method setIPAddress
  New: Method getIPAddress
  New: Method isDHCP
  New: Errorrcode -15049

- **V2015.1**
  New: Method isBTS2048VL
  New: Method isBTS2048BS
  New: Method isBTS2048UV
  New: Method isBTS2048VLTEC
  New: Method setTriggerSource
  New: Method getTriggerSource
  New: Method setTriggerInternalLevels
  New: Method setCooling
  New: Method getCoolingState
  New: Errorcodes -15051, -15053, -15054, -15055
  New: Warnings 15052, 15056
  New: Warnings 15057, 15058
  Bugfix: sometimes underloads and overloads of the integral sensor weren't detected

- **V2015.2**
  New: Method integralGetSaturation
  New: new internal calculation methods for „scale array with diode“
  New: Warnings 15057, 15058
  Bugfix: underloads and overloads of the integral sensor were not detected sometimes

- **V2015.3**
  New: Support of new measurement device types
  New: Method isBTS2048Type

- **V2015.4**
  New: additional methods for substitution
  New: additional methods for calibration
  New: Method setCooling
  New: Method hasCooling
  New: Method isBTS2048Type
  New: Method isStraylightMeasurement
  New: Method isMultiMeasurement
  New: Method spectralSetMeasurementTimeInUs
  New: Method spectralGetMeasurementTimeInUs
  New: Method getMinValidWavelength
  New: Method getMaxValidWavelength
  New: Errors -15029, -15061, -15062, -15063, -15064, -15099, -15100, -15102
  New: Warnings 15025, 15059, 15098, 15103, 15104, 15106

- **V2015.5**
  Bugfix: Performanceoptimization

- **V2015.6**
  New: Method for substitution geometry
  New: Method calibSetWavelengthMapping
  New: Method calibGetWavelengthMapping
  New: Method calibAzSetTransmissionFileActual
  New: Error -15065, -15097

- **V2015.7**
  Bugfix: Adjustments fpr saturation with substitutionMeasurementWithTestDevice and substitution↩
  MeasurementWithoutTestDevice
  Bugfix: Method getMaxADC
  Bugfix: Method getLastMaxADC

- **V2015.8**
  New: Performance-Optimization

- **V2015.9**
  New: Method calibSetCalibMode

New: Method calibGetCalibMode
New: Method integralGetLastUsedRange
New: Method spectralGetSaturation
New: Method getFilterName
New: Method setTriggerDelay
New: Method getTriggerDelay
Changed: Method getMaxADC
Changed: Method getLastMaxADC

- **V2016.1**
  New: Performance-Optimization
  New: Method spectralMeasurePremeasuredOffset
- **V2016.2**
  New: Method getLastScaleWithVLFactor
  New: Method spectralMeasureOffsetInDarkPosition
- **V2016.3**
  New: Method spectralSaveStaticOffset
  New: Method spectralLoadStaticOffset
- **V2016.6**
  New: Method spectralSetObserver10Degree
  New: Method spectralIsObserver10Degree
- **V2016.9**
  Bugfix: integral Calibration
- **V2016.10**
  New: PreciseCountCalculation
  New: spectralObserver10Degree
- **V2016.11**
  New: spectralSetAdvancedNoiseReduction
- **V2016.12**
  New: implementation of streylight matrix
  Update: spectral calibration and self-absorbtion with high resolution mode
  New: additional warnings
- **V2016.14**
  New: spectralGetLastUsedOffset
- **V2016.15**
  New: Calib HighResolutionMode
  New: Substitution HighResolutionMode
- **V2017.1**
  Bugfix: external tigger high
- **V2017.2**
  New: routine for wavelength calibration
- **V2017.3**
  Bugfix: integralGetSaturation
- **V2017.4**
  New: calibration mode with 2 Lampen (e.g. Halogen and Deuterium)
  New: integralSetIntegrationTimeInUS and integralGetIntegrationTimeInUS
  Bugfix: HighResolutionMode for standard calibration
- **V2017.5**
  Update: handling filter wheel
  Bugfix: measurement time and errors for standard BP-measurement

- **V2017.6**
  New: high resolution BP-measurement for UV-Devices
  Bugfix: Radiometric value
- **V2017.7**
  New: Export Premeasured Offset
  Update: Advanced Noise Reduction for UV Devices
  Bugfix: Multi Measurement
- **V2017.8**
  New: TM-30-15
  Update: Advanced Solar-BP Measurement
  Update: TriggerLowDuringMeasurement
- **V2017.9**
  New: CIE 170-2
  Bugfix: Selfabsorbtion Correction
- **V2017.10**
  Bugfix: loadConfigFromDevice
- **V2017.11**
  Bugfix: WL-Range for spectral calibration
- **V2018.01**
  Bugfix: setOut1LowDuringMeasurement() new FW command
- **V2018.02** - V2018.05
  New: setDHCPServer and getDHCPServer
  New: Initialisation of specific IP Adress in getHandle
  Bugfix: counts to zero after initialisation
  Bugfix: Debugger Logging Routine
  Bugfix: Interpretation of the integral status
- **V2018.06**
  New: Asynchron Measurement Methods
- **V2018.07** - V2018.10
  Update: Performance for preciseCountCalculation
  Update: Initialisation of multiple devices over LAN
  Update: time and temperature for premeasured offset
  Bugfix: measurement status for multi-measurement with high resolution mode
- **V2018.11**
  Bugfix: Cooling with LAN devices
- **V2018.12**
  Bugfix: error during first first preameasurement
- **V2018.13**
  Update: additional debugging information
  Update: Communication / TimeOut LAN
- **V2018.14** - V2018.16
  Update: FWHM, centre and centroid wavelenght for BTS2048-UV
  Update: OORSLC premeasured mode
  Update: synchronisation routine after communication timeout
  Bugfix: filterposition assignment during calibration
- **V2018.17** - V2018.19
  New: integral series measurement and output 2 low
  New: HTML documentation
  Update: calibration including linearisation correction
  Bugfix: Detection of spectral overload for NrOfScans > 1

- **V2018.20**
  New: user weighting functions
  Update: removed spectral offset in darkmode 0

Chapter 3

Errors and Warnings

A list of errors and warnings follows:

3.1 Errors

- -15000: Communication problem
- -15001: Setup file invalid for the BTS2048
- -15002: Setup file could not be opened
- -15004: az mode outside the permissible range (valid values: 0 - 2)
- -15005: Communication channel cannot be initialized
- -15006: Firmware version too low
- -15007: Problem sending file
- -15008: Problem receiving file
- -15009: BTS2048 sending an undefined error
- -15010: Delta uv limit < 0
- -15014: Error main data eeprom
- -15015: Error color data eeprom
- -15016: This command is not valid for communication per USB
- -15017: Error zero adjust integral amplifier
- -15020: Error dark current measurement
- -15026: “Exception“ received
- -15027: Filter not valid for the selected calibration table entry
- -15030: Measurement value not available since the integral measurement was not performed in the last measurement
- -15031: No values available
- -15032: Wrong password entered

- -15033: Calibration: Actual weighting function not set
- -15034: Calibration: calibration lamp spectrum not set
- -15035: Calibration: calibration name not set
- -15036: Calibration: spectral calibration factors not set
- -15037: Calibration: integral calibration factors not set
- -15038: Calibration: spectral SI unit not set
- -15039: Calibration: integral SI unit not set
- -15040: Calibration: filter assignment not set
- -15041: The wavelength range selected is too large or the step size too small resulting in a data size larger than 3300 values
- -15024: Error in technical performance pre-calculation
- -15043: Error in calculation of the CRI values
- -15044: Error in calculation of the radiometric values over the wavelength
- -15045: Error in correction with VL.
  Possible causes:
  Y = “0” -> color calculation was probably not performed
  Integral measurement value = “0” -> error in integral measurement or integral measurement not performed
- -15047: Timeout of a triggered measurement
- -15048: Selected wavelength 1 was larger or equal to the wavelength 2
- -15049: wrong format of IP-address
- -15051 Confiugration conflict (e.g. static dark value in combination with dynamic evaluation of integration time
- -15053 color calculation can’t be switched on, the defined wavelengths don’t include the complete viewable range (380nm – 780nm)
- -15054 The called method is not available for the connected measurement device
- -15055 No external power supply connected
- -15100: Parameters out of the permissible range
- -15997: No BTS2048 connected
- -15998: BTS2048 with a different serial number as the one expected connected
- -15999: Unknown error

3.2 Warnings

- 15003: File not found: no default file had been previously saved. Therefore, no default data exists
- 15011: The integral unit reports an overload
- 15012: The integral unit reports an underload
- 15023: The spectral unit reports an overload
- 15028: The integration time for the integral unit was matched to the valid grid

- 15046: If dark mode is set to static and the integration time of the spectral unit set to dynamic, the dark mode is automatically changed to dynamic since static mode is not allowed in this case.
- 15052 color calculation becomes deactivated, because the defined wavelengths don't include the complete viewable range (380nm – 780nm)
- 15056: Integration time became adapted because, the cooling was switched off
- 15057: The spectral unit reports an overload an the integral unit reports an overload
- 15058: The spectral unit reports an overload an the integral unit reports an underload
- 15094: Array low signal and integral overload
- 15095: Array low signal and integral underload

Chapter 4

Example - How to import DLL in your application

Note

The method descriptions (Module) provide examples of how to use the SDK methods.

4.1 C++

As we don't deliver import libraries for different development environments you have to use run-time dynamic linking to be able to use all methods provided by dll.
Following is an C++ example of how to import and use methods from DLL.The example does not include all available methods. The use of the handles is encapsulated in the class. The example searches and initializes a BTS2048, which is connected via USB or LAN, peforms a measurement and then records the results in the console.
At the end, all BTS2048-resources are released again.

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 More Examples

Further examples for integrating DLL´s into delphi, pyton, java, etc. can be found in the installation directory of the SDK..

Chapter 5

Module Documentation

5.1 Method information

All the methods described here can be applied to every BTS2048. Certain differences in the application can arise depending on the configuration, calibration and features of your measurement device. For instance, some methods may fail to provide any results for certain device configurations.
Each method provides a return value. Return value “0” means error-free execution of the method. Values less than “0” indicate the occurrence of an error. Values larger than “0” should be regarded as warnings.
A list of all return values is include in the documentation.

5.2 Standard SDK Methods

Methods for initialization and handlings of the SDK and the BTS2048.

Functions

- 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 Detailed Description

5.2.2 C++ Aufrufbeispiel

Following example shows the initialization of the BTS2048.

```
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 Function Documentation

5.2.3.1 GOMDBTS2048_setPassword()

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

This method has to be called before any other to activate the SDK. Activation takes place on several levels.

- level 1: general use of the SDK
- level 2: all elements of the 1st level plus saving of the calibrations in the customized memory
- level 3: all elements of level 2 plus saving calibrations in the orginal memory (recalibration)
  The passwords are separately provided to you by Gigahertz-Optik.

Parameters

in	*value*	Zero terminated string, containing the password.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.2 GOMDBTS2048_getDLLVersion()

```
int __stdcall GOMDBTS2048_getDLLVersion (
             char * value )
```

Returns the version number of this DLL.

Parameters

out	*value*	Null-terminated string; contains the version number after return, minimum size: 10 bytes.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.3 GOMDBTS2048_getDeviceList()

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

Searches for all available devices in the system.

- BTS2048 devices connected per USB return the following output: “BTS2048;Serial:<Serialnumber>;USB”
- BTS2048 devices connected per LAN return the following output: “BTS0248;Serial:<Serialnumber>;L↩ AN;IP:<IP-Address>“ <Serialnumber> is hereby the serial number of the BTS2048, <IP-Address> is the actual IP address in the network A string array must hereby be predefined for saving of the found devices. If the string array is too small, all the found devices might not be displayed. Unused array positions are marked with an empty string. The size of each of the list item is 50 characters.

Parameters

`in`	*commType*	Integer value: • -1: All devices regardless of the used communication interface • 0: Only devices connected via USB • 1: Only devices connected via LAN
`out`	*values*	String array; contains all BTS2048 devices found after return.
`out`	*listSize*	Integer; value containing the size of the device list after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.4 GOMDBTS2048_getHandle()

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

Upon activation of the SDK, this method should basically be called next in order to initialize the BTS2048. The “handle” parameter contains a unique sequence number to the instantiated measurement device that has to be passed on to the other methods as the first parameter.
After the first handle was found, the second call of getHandle(BTS2048_0, &handle), returns the next connected BTS2048, if the handle of the first device was not released meanwhile.

Parameters

`in`	*device*	Zero terminated string that identifies the device to be initialized. The string always has the following structure: “BTS2048_<serial>”. <serial> is a placeholder for the serial number of the measurement device. For example, “BTS2048_5678” initializes the BTS2048 with the serial number 5678. Another option is passing on a NULL. This initializes the first BTS2048 device registered. If you want to initialize a device over LAN with a specific IP-Adress (see setIPAdress()) you can add the Term "_IPXXX.XXX.XXX.XXX" to the initialization string (e.g. "BTS2048_0_IP192.168.002.074")
`out`	*handle*	Pointer to an integer value; this value contains a handle > 0 after return if initialization was successful and “0” if it was not

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.5 GOMDBTS2048_releaseHandle()

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

This method has to be called at the end to release the resources/memory occupied by BTS2048.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.6 GOMDBTS2048_getFirmwareVersion()

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

Returns the firmware version of the connected BTS2048.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Null-terminated string; returns the firmware version, minimum size: 10 Bytes.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.7 GOMDBTS2048_getSerialNumber()

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

Returns the serial number of the connected BTS2048.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Null-terminated string; contains the serial number of the BTS2048 after return, minimum size: 10 bytes

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.8 GOMDBTS2048_getType()

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

Returns the device type of the connected BTS2048.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Null-terminated string; contains the type of the BTS2048 after return, minimum size: 30 bytes

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.9 GOMDBTS2048_isBTS2048VL()

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

This method delivers the information, if the connected BTS2048 device is a „BTS2048-VL“ device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a boolean value; contains the information after return, if the connected device is a „BTS2048-VL“ • true: Device is a „BTS2048-VL“ device • false: Device is not a „BTS2048-VL“ device

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.10 GOMDBTS2048_isBTS2048BS()

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

This method delivers the information, if the connected BTS2048 device is a „BTS2048-BS“ device.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*value*	Pointer to a boolean value; contains the information after return, if the connected device is a „BTS2048-BS“. • true: device is a „BTS2048-BS“ device • false: device is not a „BTS2048-BS“ device

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.11 GOMDBTS2048_isBTS2048UV()

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

This method delivers the information, if the connected BTS2048 device is a „BTS2048-UV“ device.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*value*	Pointer to a boolean value; contains the information after return, if the connected device is a „BTS2048-UV“ • true: Device is a „BTS2048-UV“ device • false: Device is not a „BTS2048-UV“ device

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.12 GOMDBTS2048_isBTS2048VLTEC()

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

This method delivers the information, if the connected BTS2048 device is a „BTS2048-VLTEC“ device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a boolean value; contains the information after return, if the connected device is a „BTS2048-VLTEC“ • true: Device is a „BTS2048-VLTEC“ device • false: Device is not a „BTS2048-VLTEC“ device

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.13 GOMDBTS2048_isBTS2048Type()

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

This method delivers the information, if the connected BTS2048 is a spcified type.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*type*	Integer value; this value defines the device type, which is to be checked. 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 to an boolean value; contains the information after return, if the connected measurement device is the specified type • true: Device is the specified type • true: Device is not the specified type

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.14 GOMDBTS2048_setCooling()

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

This method enables / disables the internal cooling. The method is available only, if a BTS2048-VLTec device is connected. You need an external power supply. If cooling is enabled, then the maximum permissible spectral integration time may be up to 90 seconds. Otherwise the maximum integration time is 4 seconds.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Enable cooling • false: Disable cooling

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.15 GOMDBTS2048_getCoolingState()

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

This method tells you, if the cooling is enabled / disabled. If the cooling is enabled, the method evaulates, if the temperature is ok for stable measurements.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a integer value: • 0: Cooling disabled • 1: Cooling enabled, temperature not ok • 2: Cooling enabled, temperature ok

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.16 GOMDBTS2048_hasCooling()

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

This method returns the value, if the BTS2048 has cooling.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Parameters

out	*value*	Pointer to boolean value: • false: No cooling • true: Cooling

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.17 GOMDBTS2048_getMaxADC()

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

Returns the maximum possible number of counts of the spectral measurement unit.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the maximum possible number of counts of the spectral measurement unit after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.18 GOMDBTS2048_getNoOfPixels()

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

Returns the number of pixels of the spectral measurement unit.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the number of pixels of the spectral measurement unit after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.19 GOMDBTS2048_setIPAddress()

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

This method is setting the IP adress in the BTS2048 in the format „000.000.000.000“. Advanced zeros can be negelected. In the case of an format error (e.g. addressparts $>$ 255 or $<$ 0, more or less addressparst, wrong figures) an error code will be send out. If the IP adress „000.000.000.000“ is send, the device will be set in D←↩ HCP modus. Therby it tries to get the IP adress from the DHCP-Server during initialisation. If the device is in DHCP-modus and cannot get an IP-Adresse by the external DHCP-server, the internal DHCP-server is using the address 169.254.1.1. If the device is directly connected with an PC by LAN-cabel a address within 169.254.1.1 – 169.254.255.255 is used.
After changing the IP adress the settings are not active immediately. They are active after restarting the device. Therefore the device has to be unplugged from the power supply.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Null-terminated string; e.g. „192.168.178.25

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.20 GOMDBTS2048_getIPAddress()

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

This method is delivering the current IP-Adress.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Null-terminated String, hast o be allocated with an 16 Bytes and containts afterwards the IP address.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.21 GOMDBTS2048_isDHCP()

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

This method delivers the information if the device gets it IP adress by DHCP mode or if a fixed IP adress is choosen. Attention: This method does not tell if the DHCP Server of the device is activated or not. For this purpose use the method: isDHCPServer()

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean; gives the status: • false: Fix IP-Adresse in the device • true: IP-Adresse by DHCP

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.22 GOMDBTS2048_setDHCPServer()

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

This method sets the status of the DHCP Server of the device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	status of the DHCP Server: true = active, false = inactive

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.23 GOMDBTS2048_isDHCPServer()

```
int __stdcall GOMDBTS2048_isDHCPServer (
            int handle,
            bool * value )
```

This method delivers the information if the DHCP Server of the device is active.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean; gives the status of the DHCP Server:

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.2.3.24 GOMDBTS2048_isConnected()

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

This method checks if the device is still connected to the PC or has been disconnected

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean; gives the connection-status:

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3 Common measurement settings

Common settings for per performing a measurement.

Functions

- 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 Detailed Description

5.3.2 Function Documentation

5.3.2.1 GOMDBTS2048_saveConfig()

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

The currently set parameters are saved in a configuration file for later use. The values can be loaded using “load↩ Config”.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*filename*	zero terminated string; file name including path where the configuration data should be saved

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.2 GOMDBTS2048_loadConfig()

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

This method loads all previously set and saved values from the specified file. If the configuration file does not belong to a BTS2048 but rather to a different device, an error code is returned.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*filename*	Complete path to a configuration file where pre-existing settings are saved.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.3 GOMDBTS2048_saveConfigAsDefault()

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

The currently set parameters are saved in a configuration file for later use and are reloaded upon re-initialization of the device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.4 GOMDBTS2048_setCalibrationEntryNumber()

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

The BTS2048 is delivered with one or more calibrations. These are ideal for different measurement scenarios. This method enables you to select calibrations saved in EEPROM. A total of 52 calibration entries exist and not all have to be filled. If a non-existent calibration index is selected, the method returns an error code.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, valid value range: 0 – 51, the number of the calibration entry whose unit should be determined;

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.5 GOMDBTS2048_getSelectedCalibrationEntryNumber()

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

This method returns the selected calibration table index. This can then be used in methods such as “getUnit”. The BTS0248 is delivered with one or more calibrations. These are ideal for the different measurement scenarios. This method can be used to select calibrations saved in EEPROM. A total of 52 calibration entries exist and not all have to be filled.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*calibrationEntryNumber*	Pointer to integer value, contains the selected calibration table index after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.6 GOMDBTS2048_readCalibrationEntryInfo()

```
int __stdcall GOMDBTS2048_readCalibrationEntryInfo (
            int handle,
            int calibrationEntryNumber,
            char * calibrationName )
```

This method returns the name of the specified calibration table entry defined in the EEPROM. The BTS2048 is delivered with one or more calibrations. These are ideal for the different measurement scenarios. This method can be used to select calibrations saved in EEPROM. A total of 52 calibration entries exist and not all have to be filled. Enough memory space (max. 256 bytes) must be allocated for the calibration names before the method is called.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined;
out	*calibrationName*	Null-terminated string that contains the calibration name after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.7 GOMDBTS2048_getMeasurementQuantity()

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

This method returns the name of the stored measurement value for the specified calibration table entry. Possible return values: “E”, “I” or “Phi”.
Make sure you allocate the memory accordingly.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined;
in	*quantity*	Null-terminated string, “E”, “I”, “Phi”.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.8 GOMDBTS2048_isMeasurementQuantity()

```
int __stdcall GOMDBTS2048_isMeasurementQuantity (
             int handle,
             int calibrationEntryNumber,
             char * quantity,
             bool * isQuantity )
```

this method can be used to check if the measurement value of a specific calibration table index has a certain value. It can be checked for the variables “E”, “I” and “Phi”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined;
out	*quantity*	Null-terminated string, possible values: “E”, “I”, “Phi”
out	*isQuantity*	Pointer to a boolean value: • true: Specified calibration serves the specified measurement value • false: Specified calibration does not serve the specified measurement value

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.9 GOMDBTS2048_getSelectedMeasurementQuantity()

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

Gets the name of the currently selected calibration table entry. Possible results: “E”, “I” and “Phi”. Enough memory has to be allocated before the method is called.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*quantity*	Null-terminated string, contains the following possible values after return: “E”, “I”, “Phi”

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.10 GOMDBTS2048_isMultiMeasurement()

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

This method can be used to verify, if it is possible to use a configuration (calibration) for a multi measurment. A multi measurment is a measurement composed of several measurements in several measuring ranges.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined;
out	*value*	Pointer to boolean value: • true: Specified calibration is defined for multi measurement • false: Specified calibration is nit defined for multi measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.11 GOMDBTS2048_isOORSLCorrectionMeasurement()

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

This method can be used to check if a configuration (calibration) is an OOR (Out of Range) scattered light correction. By measuring an additional filter, the scattered light in the UV is measured and subtracted.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined;
out	*value*	Pointer to boolean value: • true: Specified calibration is defined as scattered light measurement • false: Specified calibration is not defined as scattered light measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.12 GOMDBTS2048_isSLMCorrectionMeasurement()

```
int __stdcall GOMDBTS2048_isSLMCorrectionMeasurement (
            int handle,
            int calibrationEntryNumber,
            bool * value )
```

This method returns if the configuration (calibration) is a measurement with SLM (Straylight Matrix) - correction. The scattered light is calculated on the basis of the measured spectrum and calculated by a matrix multiplication.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined;
out	*value*	Pointer to boolean value: • true: This configuration (calibration) has SLM • false: This configuration (calibration) has not SLM

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.13 GOMDBTS2048_setMeasurementMode()

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

This method defines if the measurement will be executed immediately or if the device should wait for a trigger signal. See the corresponding commands for configuration of the trigger.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*measurementMode*	integer value containing the desired mode, • 0: Immediate execution • 1: Triggered measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.14 GOMDBTS2048_getMeasurementMode()

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

This method determines the previously set measurement mode.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*measurementMode*	pointer to an integer value containing the desired measurement mode, • 0: Immediate execution • 1: Triggered measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.15 GOMDBTS2048_setSpectralIntegralSynch()

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

This method specifies if the integral and spectral measurement should be synchronously performed. If not, the spectral measurement is performed first followed by the integral measurement. Synchronous measurement is mostly recommendable if the signal is pulsed and not permanently available. Synchronous measurement also saves time.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Synchronous measurement • false: Successive measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.16 GOMDBTS2048_isSpectralIntegralSynch()

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

This method checks if synchronous measurement of the integral and spectral unit is enabled.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a boolean value: • true: Synchronous measurement • false: Successive measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.17 GOMDBTS2048_setDistance()

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

when a calibration entry for the luminous intensity or radiant intensity had been selected, the distance between the measurement device and the test object must be specified. For all the other measurement quantities, the measurement distance should be set to 1.0. Upon selection of a calibration entry that is different from “I”, the distance is automatically set to 1.0. The unit for the distance is [m].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*distance*	Double value; contains the distance in meters.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.18 GOMDBTS2048_getDistance()

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

Returns the currently defined measurement distance between the measurement device and the test object. The unit of the distance is [m].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*distance*	Pointer to a double value; contains the distance in meters.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.19 GOMDBTS2048_getFilterName()

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

Returns the name of the filter for a specific position.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*position*	Positio of filter
out	*value*	Null-terminated string, filter name (max. 39 + 1 (\0) Bytes)

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.3.2.20 GOMDBTS2048_getFilterNameforCalibration()

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

Returns the name of the filter for a specific calibration. If the calibration needs more than one filter, the method returns all filters seperated by comma.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined;
out	*value*	Null-terminated string, filter name (max. 39 + 1 (\0) Bytes)

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4 Spectral measurement setting

Measurement settings for the spectral measurement.

Functions

- 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 Detailed Description

5.4.2 Function Documentation

5.4.2.1 GOMDBTS2048_spectralSetEnabled()

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

This method activates / deactivates the spectrometer for the measurement. If the spectrometer is deactivated, no spectral measurement is performed after the next overall measurement is triggered. The spectral measurement value from the last executed measurement is thus maintained. By default, the spectrometer is activated on system start. If the spectrometer is deactivated, no dynamic a(z) correction factor is computed. This means that the last a(z) correction factor used is either maintained or that a static correction factor should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*enabled*	Boolean value: • true: Activates the spectrometer for the measurement • false: Deactivates the spectrometer for the measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.2 GOMDBTS2048_spectralIsEnabled()

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

This method checks if the spectral meter is activated for the measurement. If the spectrometer is deactivated, no spectral measurement is performed after the next overall measurement is triggered. The spectral values from the last executed measurement are thus maintained. By default, the spectrometer is activated on system start.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*enabled*	pointer to a Boolean value • true: Activated • false: Deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.3 GOMDBTS2048_spectralSetOffsetMode()

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

This method defines the offset mode used for the measurement. There are four different modes: no offset, static, dynamic and premeasured.
No offset means that no offset is subtracted from the measured signal.
Static means that the offset has to be explicitly measured once. This is done using the “spectralMeasureOffset” method. Afterwards, the measured offset is used for the measurements that follow. The “spectralDeleteOffset” method is used to reset the offset back to 0. In the case of the static offset, it should be noted that the spectral integration time can no longer be changed after the offset measurement as the offset always changes with the integration time. After changing the spectral integration time, the offset has to be measured again.
In dynamic, the offset has to be determined for each measurement. The “dark filter” is hereby used automatically and the offset measured. For the actual measurement of the wanted signal, the filter is set to the previously specified position. The filter can either be explicitly set using the “setFilterPosition” method or the filter corresponding to the selected calibration entry used.
the mode premeasured is available for measurement in dynamic time mode. the offset has to be measured for predefined integration times. this is done with the method "spectralMeasurePremeasuredOffset". The integration time of the measurement will then be adapted to one of the predefines times.
When “no offset” or “dynamic offset” is set using this method, the “spectralDeleteOffset” is then executed automatically and the last measured offset deleted.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value: • 0: No offset • 1: Static offset • 2: Dynamic offset • 3: Premeasured offset

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.4 GOMDBTS2048_spectralGetOffsetMode()

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

This method determines the previously set offset mode.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the previously defined offset mode: • 0: No offset • 1: Static offset • 2: Dynamic offset • 3: Premeasured offset

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.5 GOMDBTS2048_OORSLCorrectionSetMode()

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

This method is only available for OOR SLC (Out of Range Straylight Correction) calibration entries (only with BT←
S2048-UV devices).

With a OOR SLC calibration the straylight is meaured with a specific filter and is afterwards substracted from the measurement without filter. Each pixels is changed by a specific factor, that can be premeasured with the method OORSLCorrectionMeasureFactors(). When OORSLCMode is set to measurement of the straylight filter isn't necessary anymoreand the measurement time will be reduced./n Ths can only be done, if the spectral distribution doesn't change.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value: • 0: No premeasured factors • 1: Premeasured OOR SL-Correction factors

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.6 GOMDBTS2048_OORSLCorrectionGetMode()

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

This method determines the previously set OOR SL-Correction mode.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the previously defined offset mode: • 0: No premeasured factors • 1: Premeasured OOR SL-Correction factors

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.7 GOMDBTS2048_spectralGetIntegrationTimeRangeInus()

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

Returns the smallest and largest permissible integration time in [ms] for the spectral measurement unit of the connected BTS2048.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*min*	Ppointer to an integer value, contains the smallest permissible integration time in microseconds after return.
out	*max*	Pointer to an integer value, contains the largest permissible integration time in microseconds after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.8 GOMDBTS2048_spectralSetIntegrationTimeInus()

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

This method defines the integration time of the spectrometer. The integration times must be passed on to the method in µs.
Value range: 2 – 4000000 -$>$ 2µs to 4sec. If the integration time that is too long is specified, the spectrometer might be overloaded leading to impractical measurement results.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*timeInus*	Integer value, the integration time in µs.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.9 GOMDBTS2048_spectralGetIntegrationTimeInus()

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

This method returns the last integration time set for the spectrometer in µs.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*timeInus*	Pointer to an integer; contains the integration time in [µs] after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.10 GOMDBTS2048_spectralSetMeasurementTimeInUs()

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

This method defines the time for the whole spectral measurement (without dark measurment). For most calibration entries, measurement time and integration time are equal and both methods (this and "spectralSetIntegration↩ TimeInus()") can be used the same way. With the BTS2048-UV devices there are some calibration entries, that combine measuremnts with different filter positions.In this case the integration time defines the duration for one filter and the measurement time defines the duration for the whole measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value; measurement time in µs.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.11 GOMDBTS2048_spectralGetMeasurementTimeInUs()

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

This method returns the last set measurment time [µs](multi measurement) for the spectrometer.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to integer; integration time in [µs] after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.12 GOMDBTS2048_spectralSetDynamicTimeMode()

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

this method specifies whether the integration time for the spectral measurement unit should be dynamically determined for each measurement. If activated, the device performs a pre-measurement before the actual measurement. The actual integration time used can be retrieved after the measurement using “spectralGetIntegrationTimeInus”. Dynamic determination of the integration time is not compatible with the “static dark mode”. If the dynamic mode is activated, the dark mode is automatically changed to “dynamic”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • false: Dynamic integration time determination deactivated • true: Dynamic integration time determination activated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.13 GOMDBTS2048_spectralGetDynamicTimeMode()

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

This method checks whether the integration time for the spectral measurement unit should be dynamically determined for every measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean: • false: Dynamic integration time determination deactivated • true: Dynamic integration time determination activated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.14 GOMDBTS2048_spectralSetMaxIntegrationTimeInUs()

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

This method sets the maximal integration time. In dynamic time mode it is guaranteed, that the maximal integration time won't be exeeded.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*value*	Double value, max. integration time in µs

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.15 GOMDBTS2048_spectralGetMaxIntegrationTimeInUs()

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

This method returns the current value for the maximal integration time.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*value*	Pointer to double value, returns the max. integration time in µs.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.16 GOMDBTS2048_spectralSetMaxMeasurementTimeInUs()

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

This method sets the maximal measurement time. In dynamic time mode it is guaranteed, that the maximal measurment time won't be exeeded. For the difference between integration time and measurment time look the description of spectralSetMeasurementTimeInUs

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*value*	Double value, max. measurement time in µs

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.17 GOMDBTS2048_spectralGetMaxMeasurementTimeInUs()

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

This method returns the current value for the maximal measurement time.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to double value, returns the max. measurement time in µs.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.18 GOMDBTS2048_spectralSetNrOfScans()

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

This method defines the number of averagings of your spectral measurements. The spectral measurements will thus be performed correspondingly often and the results averaged. This leads to improved measurement results but worsens the system performance. This setting must be made before the “measure” method is called. The default value after initialization is “1”.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*nrOfScans*	Integer value, contains the number of spectral measurements‘ that should be averaged per measurement.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.19 GOMDBTS2048_spectralGetNrOfScans()

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

This method returns the number of spectral measurements previously set for the averaging for the spectrometer.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*nrOfScans*	Pointer to integer value; contains the number of averagings.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.20 GOMDBTS2048_setWavelengthRange()

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

This method defines the constraining wavelength ranges for subsequent calls of the “spectralGetCountsWavelength” or “spectralGetSpectrumCalibratedWavelength” methods. The wavelength range also has influence on the calculation of the half width that can be obtained through “getFWHM”.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*L1*	Double value, minimum wavelength in [nm]
`in`	*L1*	Double value, minimum wavelength in [nm]
`in`	*L2*	Double value, maximum wavelength in [nm]
`in`	*dL*	Double value, step size in [nm]

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.21 GOMDBTS2048_getWavelengthRange()

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

returns the previously set wavelength range that was used in the other methods e.g., “spectralGetCounts↩ Wavelength” or “spectralGetSpectrumCalibratedWavelength”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*L1*	Pointer to a double value, contains the minimum wavelength in [nm]
out	*L2*	Pointer to a double value, contains the maximum wavelength in [nm]
out	*dL*	Pointer to a double value, contains the step size in [nm]

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.22 GOMDBTS2048_getMinValidWavelength()

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

This method returns the min. valid wave lengh of the current selcted calibration entry.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to double value, returns the min. valid wave lengh [nm]

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.4.2.23 GOMDBTS2048_getMaxValidWavelength()

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

This method returns the max. valid wave lengh of the current selcted calibration entry.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to double value, returns the max. valid wave lengh [nm]

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5 Spectral correction methods & filter

Functions

- 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 Detailed Description

5.5.2 Function Documentation

5.5.2.1 GOMDBTS2048_spectralSetScaleWithIntegralMode()

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

This method is used to define the scaling mode of the spectral funktion. If scaling is activated the spectral funktion gets scaled so the radiometric value matches with the value of integral detector. There are three modes available:

- 0: None -> Scaling deactivated
- 1: Always on -> Scaling always activated
- 2: Automatic -> Scaling activated if recommended. This means the spectrum is only scaled if the integral detektor has enough signal, AZ-Mode is dynamic or automatic and the scaling facor doesn't change the signal by more then 20%.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integral value: • 0: None • 1: Always on • 2: Automatic

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.2 GOMDBTS2048_spectralGetScaleWithIntegralMode()

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

Returns the actual scaling mode of the spectral funktion

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to Integral value: • 0: None • 1: Always on • 2: Automatic

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.3 GOMDBTS2048_spectralSetScaleWithVLambda()

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

This method is used to activate and deactivate scaling. If activated, spectral data is absolutely scaled using the integral sensor.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Scaling activated • false: Scaling deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.4 GOMDBTS2048_spectralIsScaleWithVLambda()

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

Checks if scaling is activated or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to a Boolean value: • true: Scaling activated • false: Scaling deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.5 GOMDBTS2048_spectralSetPixelLinearization()

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

This method is used to activate and deactivate pixel linearization. If activated, it causes linearization of the spectral sensor.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Linearization activated • false: Linearization deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.6 GOMDBTS2048_spectralIsPixelLinearization()

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

Checks whether pixel linearization is activated or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a boolean value: • true: Linearization activated • false: Linearization deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.7 GOMDBTS2048_spectralSetBandwidthCorrection()

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

this method activates and deactivates bandwidth correction. The bandwidth correction is based on a fitting method from Woolliams which is recommended by CIE TC2.51.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Correction activated • false: Correction deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.8 GOMDBTS2048_spectralIsBandwidthCorrection()

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

Checks whether bandwidth correction is activated or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to a Boolean value: • true: Correction activated • false: Correction deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.9 GOMDBTS2048_spectralSetSavitzkyGolayFilter()

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

With this methode the noise reduction accordiont to an Savitzky-Golay filter can be activated or deactivated. This algortihmus cannot be used at the simultaneously tot he „spectralSetNoiseReduction” methode. By activated this method the other one will be deactivated automatically.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • true: Savitzky golaf filter active • false: Savitzky golaf filter inactive

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.10 GOMDBTS2048_spectralIsSavitzkyGolayFilter()

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

Checks if the noise reduction with SavitzkyGolayFilter is active.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean value: • true: Active • false: Inactive

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.11 GOMDBTS2048_spectralSetNoiseReduction()

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

This method activates and deactivates noise reduction of the spectral sensor. In this method, +-2 adjacent pixels are averaged. This is admissible since these pixels are still within the bandwidth of the device. This algorithm is recommended for noisy signals since for a simple measurement, a 5-times averaging is performed. This is particularly recommended for broadband light sources. For linear lamps or lasers, this algorithm results in a bandwidth extension and is hence not the optimal choice.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value • true: Noise reduction activated • false: Noise reduction deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.12 GOMDBTS2048_spectralIsNoiseReduction()

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

Checks whether noise reduction of the spectral sensor is activated or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a boolean value: • true: Noise reduction activated • false: Noise reduction deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.13 GOMDBTS2048_spectralSetDarkThreshold()

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

The dark threshold defines the minimum number of counts required for the signal to be analyzed and processed at the respective pixel.
Value range: 0 – 65535.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value; number of counts that must be exceeded as the dark threshold. Value range: 0 - 65535

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.14 GOMDBTS2048_spectralGetDarkThreshold()

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

Returns the number of counts defined as the dark threshold. Valid value range: 0 – 65535.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value; number of counts defined as the dark threshold.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.15 GOMDBTS2048_spectralSetObserver10Degree()

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

This method defines the CIE Observer for the color calculation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • false: CIE 1931 observer with 2° range of vision • true: CIE 1964 observer with 10° range of vision

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.16 GOMDBTS2048_spectralIsObserver10Degree()

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

This method returns the CIE Observer for the color calculation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean value: • false: CIE 1931 observer with 2° range of vision • true: CIE 1964 observer with 10° range of vision

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.17 GOMDBTS2048_spectralSetAdvancedNoiseReduction()

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

This method turns on the improved noise reduction. The noise reduction is a specially developed filter, which smoothes the spectrum dynamically and suppresses noise.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • false: Not active • true: Active

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.18 GOMDBTS2048_spectralIsAdvancedNoiseReduction()

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

This method returns if the improved noise reduction is active or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean value: • false: Not active • true: Active

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.19 GOMDBTS2048_setPreciseCountCalculation()

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

This method defines the calculation for the counts. It´s possible to use integer values or floating point values. The floating point values are more precise, but the performance is less than the integer values.
Especially if the number of averages "NrOfScans" is set more than 1. In this case it helps to use PreciseCount↩
Calculation().

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • false: Low Accurancy of integer • true: High Accurancy of floating point

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.5.2.20 GOMDBTS2048_getPreciseCountCalculation()

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

This method returns which accurancy for calculation of the counts is used (integer oder floating point).

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*value*	Boolean value: • false: Low Accurancy of integer • true: High Accurancy of floating point

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6 Integral measurement settings

Functions

- 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 Detailed Description

5.6.2 Function Documentation

5.6.2.1 GOMDBTS2048_integralSetEnabled()

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

this method activates / deactivates the integral sensor for the measurement. If the integral sensor is deactivated, no integral measurement is performed after the next general measurement is triggered. The integral measurement value from the last executed measurement is thus maintained. By default, the integral sensor is activated on system start.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*enabled*	Boolean value: • true: Activates the integral sensor for the measurement • false: Deactivates the integral sensor for the measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.2 GOMDBTS2048_integralIsEnabled()

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

This method checks if the integral sensor is activated for the measurement. If the integral sensor is deactivated, no integral measurement is performed after the next overall measurement is triggered. The integral measurement value from the last executed measurement is thus maintained. By default, the integral sensor is activated on system start.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*enabled*	Pointer to boolean value: • true: Activates • false: Deactivates

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.3 GOMDBTS2048_integralGetIntegrationTimeRangeInMs()

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

Returns the smallest and largest permissible integration time in [ms] for the integral measurement unit of the connected BTS2048.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*min*	Pointer to an integer value, contains the smallest permissible integration time milliseconds.
out	*max*	Pointer to an integer value, contains the largest permissible integration time milliseconds.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.4 GOMDBTS2048_integralSetIntegrationTimeInUs()

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

This method is used to specify the integration time for the integral sensor. The integration times must be passed on to the method in [us].
Value range by flicker-devices with fast integral amplifier: 20 – 2000000 -> 20 us to 2 s in 20us steps.
Value range with a normal BTS2048 device: 1000 – 2000000 -> 1 ms to 2 s.
There is a specific integration time for the 0 – 5 measurement range (parameter range = 0) and another for the 6 – 8 measurement range (parameter range = 1).
After setting a new integration time, a “zero measurement” that requires extra processing time depending on the selected measurement range is automatically performed. This consists of the actual integration time for the “zero measurement” and the waiting time before the start of the measurement (currently 500ms for the 0 – 5 measurement range and 1000ms for the 6 – 8 measurement range).
If the newly set time is identical to the already activated integration time, the zero measurement is not performed.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*range*	Integer value: • 0: For the 0 – 5 measurement range • 1: For the 6 – 8 measurement range
`in`	*value*	Integer value, the integration time microseconds [us].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.5 GOMDBTS2048_integralGetIntegrationTimeInUs()

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

This method returns the last seted integration time for the selected range of the integral sensor. The unti for the time is us.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*range*	Integer value: • 0: Range 0 – 5 • 1: Range 6 - 8
`out`	*time*	Pointer to integer value; integration time in microsecond [us].

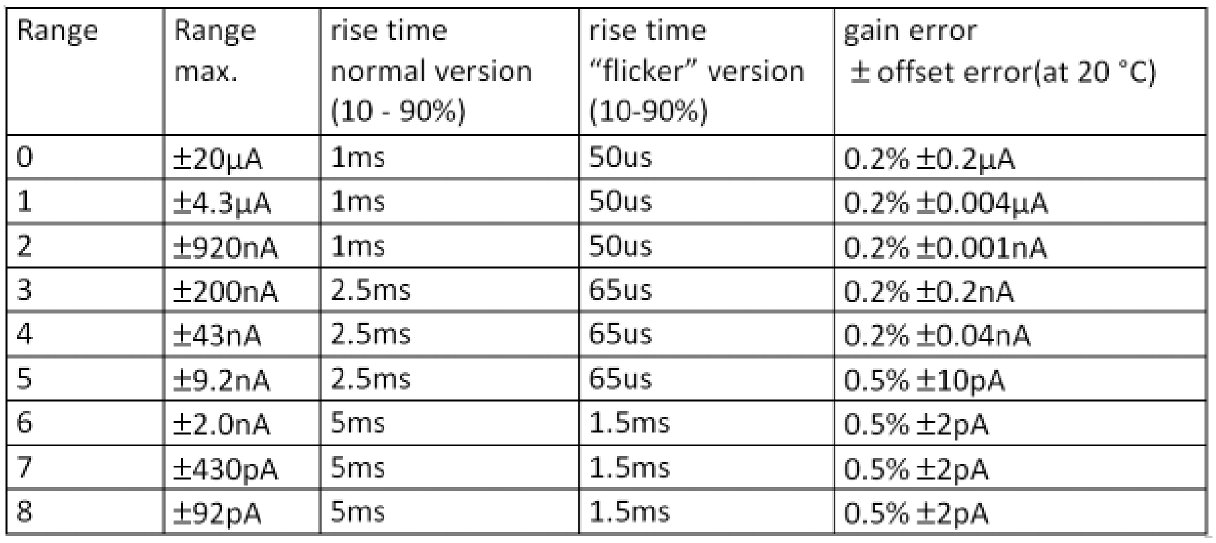
Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.6 GOMDBTS2048_setIntegralRange()

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

defines the measurement range of the signal to be measured.



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 means auto-ranging. The device independently searches for the optimum measurement range for the signal to be measured.
The automatic switching can lead to undefined results in triggered measurements and has a negative effect on the performance.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	integer value; contains the measurement range: • -1: Auto ranging • 0 – 8: Specific measurement range (0 = insensitive, 8 = sensitive)

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.7 GOMDBTS2048_getIntegralRange()

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

Returns the currently set measurement range.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value; contains the currently set measurement range: • -1: auto ranging • 0: insensitive • ... • 8: sensitive

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.8 GOMDBTS2048_integralSetRangeWaitTimeInMs()

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

This method can be used to set the delay time for switching the integral range range. The unit of the wait time is ms. For range 0 it is possible to have a wait time of 5 ms to 20 ms. For range 1 it is possible to have a wait time of 20 ms to 200 ms.
Default value:
rangeArea 0: value = 20 ms
rangeArea 1: value = 200 ms
The default values should not be overwritten. Only your application needs a better performance.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*rangeArea*	Integer value: • 0: Measurement range 0 – 5 • 1: Measurement range 6 – 8
in	*value*	die Wartezeit in Millisekunden.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.9 GOMDBTS2048_integralGetRangeWaitTimeInMs()

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

This method returns the last seted wait time for the integral sensor. It is the value, if the devices switches to another measurement range.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*rangeArea*	Integer value: • 0: Measurement range 0 – 5 • 1: Measurement range 6 – 8
out	*value*	Pointer to integer, wait time in ms.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.10 GOMDBTS2048_integralSetAzMode()

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

This method defines the a(z) correction mode. The a(z) correction is used for spectral matching of the integral sensor. The following options exist:

- mode = 0: no spectral matching
- mode = 1: spectral matching with a static correction factor that can be defined using the “integralSetAz←Specific” method
- mode = 2: dynamic spectral matching. A new a(z) correction factor is computed from the spectrum after every measurement; spectral measurement has to be activated in this case.
- mode = 3: automatic spectral matching. A new a(z) correction factor is computed as in mode 2 but only if the signal in relevant spectral range is high enough. Otherwise a(z) is 1; spectral measurement has to be activated in this case.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*mode*	Integer value of the a(z) mode to be set.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.6.2.11 GOMDBTS2048_integralGetAzMode()

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

This method returns the previously defined a(z) correction mode. The following modes exist:

- mode = 0: no a(z) correction
- mode = 1: correction with a static factor that can be computer using the "integralSetAzSpecific" method.
- mode = 2: correction with a dynamically computed a(z) correction factor based on a previously performed spectral measurement.
- mode = 3: correction with a automatic computed a(z) correction factor based on a previously performed spectral measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*mode*	Pointer to integer value; returns the a(z) mode

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.6.2.12 GOMDBTS2048_integralSetAzSpecific()

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

Defines a static a(z) correction factor. The correction factor is used for spectral matching of the integral sensor. This value is only applied if the a(z) mode is set to "1".

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*az*	Double value representing the a(z) correction factor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.6.2.13 GOMDBTS2048_integralGetAzSpecific()

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

Returns the previously defined static a(z) correction factor. The correction factor is used for spectral matching of the integral sensor. This value is only applied if the a(z) mode is set to “1”.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*az*	pointer to a double value; contains the static a(z) correction factor after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7 Methods for color calculation

Functions

- 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 Detailed Description

5.7.2 Function Documentation

5.7.2.1 GOMDBTS2048_setColorCalculation()

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

Color values will be calculated after the spectral measurement only if “color calculation” is activated; these can then be fetched using “getColor”. If color calculation is not necessary, this option can be deactivated to save up on time.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value that ascertains if the color calculation should be activated: • true: Color calculation activated • false: Color calculation deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.2 GOMDBTS2048_isColorCalculation()

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

Checks if the color calculation is activated or not. Color values will be calculated after the spectral measurement only if color calculation is activated; these can then be fetched using “getColor”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to boolean: • true: Color calculation active • false: Color calculation not active

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.3 GOMDBTS2048_setColorCalculationMode()

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

The color can be calculated based on the individual pixels or the predefined wavelength range with the focus points at a distance equal to the step size. The calculation based on the pixels is the most accurate option but may take slightly longer compared to calculation based on focus points in the wavelength range.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value; contains the mode: • 0 : Pixel-based • 1: Wavelength-based

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.4 GOMDBTS2048_getColorCalculationMode()

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

Determines the mode in which the color calculations are performed. 0 – pixel-based, 1 – wavelength-based.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Parameters

out	*value*	Pointer to an integer value; color calculation mode: • 0 : Pixel-based • 1: Wavelength-based

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.5 GOMDBTS2048_setColorCalculationOptimizationMode()

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

Certain preliminary calculations are necessary for calculation of color data. These preliminary calculations are very time-consuming. It has therefore been made possible to save the results of the preliminary calculations for both the pixel-based and wavelength-based calculation in the internal memory. This saves time during computation of the color data.
In order to load both the pixel-based and wavelength-based data, this pre-calculation has to be done for one of the different data types after the measurement.
Value range: 0 – no pre-calculation, 1 – pixel-based, 2 – wavelength-based
The wavelength-based pre-calculation is done, if activated, each time the wavelength range is changed.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value; contains the mode: • 0: Inactive • 1: Pixel-based • 2: Wavelength-based

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.6 GOMDBTS2048_getColorCalculationOptimizationMode()

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

Checks whether or not the calculation optimization is activated.

- 0: optimization not activated
- 1: optimization for pixel-based calculation activated
- 2: optimization for wavelength-based calculation activated

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*value*	Pointer to an integer value; color calculation optimization: • 0: inactive • 1: pixel-based, • 2: wavelength-based

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.7 GOMDBTS2048_setCTLimitCheckActive()

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

The color temperature is only calculated with respect to the uv limits only when the CT limit check is activated. This method activates and deactivates the CT limit check.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*value*	Boolean value ascertains if the limit check is activated or not: • true: Limit check activated • false: Limit check deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.8 GOMDBTS2048_isCTLimitCheckActive()

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

Checks if the CT limit check is activated. The color temperature is calculated depending on whether the limit check is activated or not and with respect to the defined uv limit.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to a Boolean value, after return, it contains information on whether the CT limit check is activated: true: CT limit check activated false: CT limit check deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.9 GOMDBTS2048_setDeltaUVLimit()

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

Defines the acceptable uv limits within which calculation of the color temperature should be done. If the actual values are outside the limits, the color temperature is not calculated. The default value is “0.05“. The limit is activated and deactivated in combination with the “setCTLimitCheckActive” method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*limit*	Double value, the limit for calculation of the color temperature

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.7.2.10 GOMDBTS2048_getDeltaUVLimit()

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

Returns the previously set uv limit used for calculation of the color temperature. The limit is only applicable if the method “setCTLimitCheckActive” is activated

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*limit*	Pointer to a double value, contains the defined limit after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8 Measurement methods

methods for preparing and performing a measurement

Functions

- 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 Detailed Description

5.8.2 Function Documentation

5.8.2.1 GOMDBTS2048_measure()

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

This method triggers the measurement. It uses the previously specified settings e.g., integration time, activation of the measurement sensors, calculation of color values . . .
After the measurement, wanted measurement results can be read out from the device using the corresponding methods.
If one is in the "triggered measurement" mode, the "isMeasurementFinished" method has to be called repeatedly until it returns "true". Other methods may be called only after this.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.2 GOMDBTS2048_spectralEvaluateIntegrationTimeInus()

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

This method performs a test measurement and returns the optimal integration time in us for the spectrometer with the currently set filter.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*timeInus*	Pointer to integer value; contains the integration time in [us] after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.3 GOMDBTS2048_measureGetCountsPixelFast()

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

This method is used to trigger ultra-fast measurements with a fast data-readout process without considering offset values. It triggers a spectral measurement with the set integration time. All other parameters used for standard measurements are not taken into consideration in this method. After the measurement, the spectral unit counts are automatically read. Depending on the technical environment (LAN, WLAN, PC, etc.) and the integration time, up to 5ms can be attained when this method is used for the measurement and data readout process. This method is not available for USB mode and always returns an error code in this case.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to the first element of a double array, contains the counts for each pixel after return, the array requires enough memory for 2048 double values.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

Example:

```
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;
```

5.8.2.4 GOMDBTS2048_spectralMeasureOffset()

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

This method is used to measure the offset. The currently set filter position and the integration time are hereby used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

Example:

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

5.8.2.5 GOMDBTS2048_spectralSaveStaticOffset()

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

With this method, you can store the static offset for a later measurement. In this case the dark counts and integration time will be stored. It is possible to store as many measurements as you want. With releaseHandle() all offset measurements are deleted and the memory is released.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.6 GOMDBTS2048_spectralLoadStaticOffset()

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

This method loads a previously stored static offset. If no offset has been stored, the method returns the error code -15076.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.7 GOMDBTS2048_spectralMeasureOffsetInDarkPosition()

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

It´s like spectralMeasureOffset(), but the offset is measured by dark position of the filter wheel. After that, the filter goes to the previous position.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.8 GOMDBTS2048_spectralMeasurePremeasuredOffset()

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

With this method, you can measure the "premeasured offset" for some internal defined times. You need this, for the premeasured offset mode. The dark position of the filter is always used. After the measurement the filter is goingt to the previous position. The function call takes time, for example the BTS2048-VL needs 15 seconds and the BTS2048-UV need 120 seconds.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.9 GOMDBTS2048_spectralExportPremeasuredOffset()

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

With this method, you can save the "premeasured offset" for later usage in a file. But it is recommended to measure a new offset from time to time

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*filename*	Zero terminated string, contains the complete path to the file to be loaded.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.10 GOMDBTS2048_spectralImportPremeasuredOffset()

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

With this method, you can load the "premeasured offset" from a file on your disk.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*filename*	Zero terminated string, contains the complete path to the file to be loaded.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.11 GOMDBTS2048_spectralDeleteOffset()

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

This method is used to reset the current offset in the device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

Aufrufbeispiel:

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

5.8.2.12 GOMDBTS2048_OORSLCorrectionMeasureFactors()

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

This method is only available for OOR SLC (Out of Range Straylight Correction) calibration entries (only with BT←S2048-UV devices).
It measures the OOR SL-Correction factors, that can be used with the method OORSLCorrectionSetMode()

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.13 GOMDBTS2048_integralMeasureOffset()

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

With this method you can measure the integral offset. It is used the current settings of filter position and integration time.
Attention:
If you want to measure the dark current, you have to set the position of the filter wheel on "dark" = position 0. Better is to use the method integralMeasureOffsetInDarkPosition().

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.14 GOMDBTS2048_integralMeasureOffsetInDarkPosition()

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

With this method you can measure the integral offset, if the filter wheel is in dark position. Afterwards the filter goes to the original position back. The integral offset is measured by initalisation of the BTS2048 device. It is only necessary to measure the integral offset, if the signal is very week. The offset is independently of the integral measureing time.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.15 GOMDBTS2048_setFilterPosition()

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

Your *BTS2048-VL*: is featured with a filter wheel containing up to 4 different filters for signal damping.
Value range: 0 – 3, with:

- 0: blocking filter
  - 1: OD2
  - 2: OD1
  - 3: no filter Your *BTS2048-UV*:: is featured with a filter wheel containing up to 4 different filters for signal damping.
    Value range: 0 – 7, with:
  - 0: blocking filter
  - 1: OD2

- 2: OD1
- 3: no filter
- 2: OoR filter
- 3: BP filter 1
- 4: BP filter 2
- 5: BP filter 3
- 6: BP filter 4
- 7: BP filter 5

This filter position is used for the measurement and corresponds to the filter position defined in the calibration entry.
If the set filter does not match the selected calibration entry, only the counts can be read out as the calibrated values will thus be invalid.
If a new calibration entry is selected, the filter position in the calibration entry is used for the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*position*	Integer value; contains the filter number that should be used.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.16 GOMDBTS2048_getFilterPosition()

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

your BTS2048 is featured with a filter wheel containing up to 4 (BTS2048-VL) or 8 (BTS2048-UV) different filters for damping the signal (see OMDBTS2048_setFilterPosition()). This method returns the currently set position of the filter wheel.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*position*	Pointer to an integer value; contains the currently set filter position number.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.17 GOMDBTS2048_integralSeriesMeasure()

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

With this method a fast series of integral measurements will be performed. The number of measurements has to be defined by the parameter count and each measurement has to be started with a single trigger pulse. Trigger Input has to be set to 1 or 3 and integral range has to be set to a fixed value. (no autorange) Before calling this function the trigger settings (high/low, level/edge) have to be defined with the methods setTriggerMode() and setTriggerLevel(). After all measurements have been done this method will return and the measurement values can be collected using the method integralGetSeriesValues(). Keep in mind that the duration between the trigger signals has to be larger than the integral integration time

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*count*	integer value; number of measurements that should be done.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.18 GOMDBTS2048_integralGetSeriesValues()

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

This method returns the measurement values, that have been recorded with the function integralSeriesMeasure().

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*values*	pointer to the first element of a double array, contains the measurement values, the size of the array must be at least of the size of measurements predefined in integralSeriesMeasure()

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.19 GOMDBTS2048_setOut2LowDuringIntegralSeriesMeasurement()

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

This method sets the output 2 of the device to low during a measurement of the integral series takes place.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	boolean value: false: off, true: output 2 low during measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.8.2.20 GOMDBTS2048_getOut2LowDuringIntegralSeriesMeasurement()

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

This method returns checks the status of the output 2 low during integral series measurement

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	pointer to boolean value: false: off, true: output 2 low during measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.9 Asynchron measurement methods

methods for performing an asynchron measurement

Functions

- 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 Detailed Description

5.9.2 Function Documentation

5.9.2.1 GOMDBTS2048_asyncStartMeasurement()

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

This method triggers an asynchronous measurement. It does exactly the same as the methode "measure()" but will return immediately after call and the measurement will be performed in a seperate background thread. When using this method, basic know-how about multithreading is assumed. The progress of the measurement can be fetched using the method asyncGetProgress(). It is designated to call the asyncGetProgress() method repeatedly, until the finished parameter is "true". The backgroung thread and the main thread will then be joined automatically.
The asnchron measurement is not compatible with the triggered measurement. After the measurement is done, results can be read out from the device using the corresponding methods.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.9.2.2 GOMDBTS2048_asyncStartMeasurementWithTime()

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

This method also starts an asynchronous measurement, but when called it additionally returns the time the measurement is expected to take. For this reason the method doesn't return immediately. Measurement time is calculated first and when dynamicTimeMode is activated, the premeasurement to determine the integration time is done before the method returns.

Keep in mind, that there are some situations, when the measurement time can not be calculated before actually performing the measurement. (e.g. with multi-measurements with the BTS2048-UV) In this case the time should be considered as an assumption.
The handling of this method is the same as for asyncStartMeasurement().

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*time*	pointer to double value; contains the estimated measuremnt time in seconds.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.9.2.3 GOMDBTS2048_asyncGetProgress()

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

During an asynchronous measurement with this method, the status of the measurement can be fetched. Additionally it joins the background thread of the measurement with the main thread. For this reason it is absolutely neccessary to call this method until the parameter finished returns at least once the value "true"

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*finished*	pointer to boolean value; is the measurement finished?
out	*progress*	pointer to integer value; progress in %. Will be updated sporadic.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.9.2.4 GOMDBTS2048_asyncStopMeasurement()

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

With this method an asynchronous measurement can be stopped before the measurement is actually finished. But this method has no impact on the measurement process of the device, for this reason it can take some time until the measurement is actually canceled.
After using this method it is also absolutely neccessary to call the method asyncGetProgress() until the parameter finished returns "true". This way you will get the status of the measurment abruption.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.10 Methods for reading integral measurment values

mehtods for collecting the previously measured integral values

Functions

- 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 Detailed Description

5.10.2 Function Documentation

5.10.2.1 GOMDBTS2048_integralGetUnit()

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

gets the SI unit of integral sensor's calibration. A total of 52 calibration entries exist and not all have to be filled. Calibration entries that are empty return an empty string. Enough memory (max. length: 20 bytes) has to be allocated for the return string.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined; valid value range: 0 – 51
out	*unit*	Zero terminated string; contains the SI unit after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.10.2.2 GOMDBTS2048_integralGetSaturation()

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

gets the saturation of the integral unit within the selected range of the previous measurement. The method returns the result as a value in percent and a value based on the maximum of a 15bit number. 0% equates to the value „0“. 100% equates to the value „32768“.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*saturationPercent*	Pointer to double, contains the saturation in percent after return
in	*saturation15bit*	Pointer to double, contains the saturation; reference value: 100% = 32768

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.10.2.3 GOMDBTS2048_integralGetLastUsedAz()

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

returns the a(z) correction factor that was last used. This factor is either computed based on the last measured spectrum, statically defined or set to 1.0 if the a(z) mode is set to “0”. The correction factor is used for spectral matching of the integral sensor.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*az*	Pointer to a double value, contains the a(z) correction factor after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.10.2.4 GOMDBTS2048_integralGetValue()

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

This method returns the integral measurement value corrected with the calibration factor and a(z) correction factor. This basically involves the illuminance, luminous intensity or luminous flux depending on the system and the selected calibration table entry.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, measurement value of the integral sensor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.10.2.5 GOMDBTS2048_integralGetCurrent()

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

This method returns the raw value (current in [A]) of the integral measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, measured current of the integral sensor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.10.2.6 GOMDBTS2048_integralGetLastUsedRange()

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

This method returns the integral range used during the last measurment. This is helpful, if autorange is activated.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*range*	Pointer to a double value, last used integral range.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11 Methods for reading spectral measurment values

mehtods for collecting the previously measured spectral values

Functions

- 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 Detailed Description

5.11.2 Function Documentation

5.11.2.1 GOMDBTS2048_spectralGetUnit()

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

Gets the SI unit of the spectral sensor's calibration. A total of 52 entries exist and not all have to be filled. Calibration entries that are empty return empty strings. Enough memory (max length: 20 bytes) has to be allocated for the return string.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*calibrationEntryNumber*	Integer value, the number of the calibration entry whose unit should be determined; valid value range: 0 – 51.
out	*unit*	Zero terminated string; contains the SI unit after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.11.2.2 GOMDBTS2048_spectralGetSaturation()

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

This method returns the spectral saturation of the last

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*saturation*	Pointer to integer value, contains saturation after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.3 GOMDBTS2048_getRadiometricValueOverWLRange()

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

This method returns the radiometric value by calculating an integral value over the defined wavelength range (set↩ WavelengthRange()).

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, contains “large” X.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.4 GOMDBTS2048_getPeak()

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

: returns the peak value of the spectral measurement. Both the measurement value and the corresponding wavelength in [nm] are hereby determined. The spectral unit depends on the selected calibration table entry and can be determined using the “spectralGetUnit” method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*lambda*	Pointer to a double value, contains the wavelength in [nm].
out	*power*	Pointer to a double value, contains the peak measurement value.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.5 GOMDBTS2048_getFWHM()

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

Returns the full width at half maximum (FWHM = full width at half maximum) of the measured spectrum.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*fwhm*	Pointer to a double value, contains the full width at half maximum value in [nm].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.6 GOMDBTS2048_getCenterWavelength()

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

Returns the midpoint of the FWHM in [nm].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, contains the midpoint of the FWHM in [nm].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.7 GOMDBTS2048_getCentroidWavelength()

```
int __stdcall GOMDBTS2048_getCentroidWavelength (
            int handle,
            double * value )
```

returns the centroid wavelength. The centroid wavelength is a measure used to characterize a spectrum. It indicates the "center" of the spectrum.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, contains the centroid wavelength in [nm].

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.11.2.8 GOMDBTS2048_spectralGetSpectrumCalibratedWavelength()

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

Returns the results of measurement with the spectrometer using the calibration and substitution factors. The expected number of measurement values should be predefined using "spectralGetSpecmax" in order to allocate enough memory for the results array.
The first element of the results contains a measurement value for the defined start wavelength.
The second element of the results contains a measurement value for the defined start wavelength + defined step size . . . .
The interpolation points are defined using the "setWavelengthRange" with which the start wavelength, end wavelength and step size can be defined.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*spectrum*	pointer to the first element of a double array, contains the calculated measurement values, the size of the array must be predefined and is dependent on the previously defined wavelength range and step size

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.11.2.9 GOMDBTS2048_spectralGetSpectrumCalibratedPixel()

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

Returns the the measurement result with calibration factor and substitution factor. 2048 values are returned corresponding to the 2048 pixels of the array.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*spectrum*	Pointer to a double array; contains the calculated measurement results after return; the size of the array are 2048 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.10 GOMDBTS2048_spectralGetCountsPixel()

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

returns the raw values of the measured spectrum. These are based on the single counts for each pixel. A 2048 large double array must be available for the values to be returned in.

- The first element of the results contains the counts of the first pixel
- The second element of the results contains the counts of the second pixel
- ...

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*counts*	pointer to the first element of a double array, contains the counts for each pixel after return, the array requires enough memory size for 2048 double values.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.11 GOMDBTS2048_spectralGetLambdas()

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

Returns the wavelengths.
If the “wavelengthRaster” is set to true, the wavelengths of the set wavelength range are returned. The size of the

double array is dependent on the start wavelength, end wavelength and step size and can be determined using “spectralGetSpecmax”.
If “wavelengthRaster” = false, one gets back the wavelength allocation for the 2048 pixels of the spectral measurement unit. A 2048 large double array must hereby be available for the results to be return in.

- The first element of the results contains the “wavelength” of the first pixel
- The second element of the results contains the “wavelength” of the second pixel
- ...

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*wavelengthRaster*	Boolean value; • true: Wave lengh refer to „WavelengthRange“ • false: Wave lengh assignment to the 2048 pixwl of the spectral measurement device
out	*lambdas*	Pointer to the first element of a double array, contains the counts for each pixel after return, the array requires enough memory for 2048 double values.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.12 GOMDBTS2048_spectralGetSpecmax()

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

This method returns the number of elements to be expected when the “spectralGetCountsWavelength” method is called. The number varies depending on the defined wavelength range and step size.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the number of elements to be expected.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.11.2.13 GOMDBTS2048_spectralGetLastUsedOffset()

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

Returns the dark counts of the last executed spectral measurement. If the values are exist.

Parameters

`in`	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*values*	Pointer of the first element of a double array, contains the dark counts after return, min. size 2048 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.12 Methods for reading common measurment values

Functions

- 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 Detailed Description

5.12.2 Function Documentation

5.12.2.1 GOMDBTS2048_getTemperature()

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

Measures the actual temperature of the temperature sensor. The unit used is [℃].

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value; contains the temperature in [℃].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.12.2.2 GOMDBTS2048_getLastMaxADC()

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

Returns the maximum count of the last spectral measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the maximum counts of the last spectral measurement.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.12.2.3 GOMDBTS2048_getLastScaleWithVLFactor()

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

Returns the scaling factor of the last spectral measurement. For more information look the method setScaleWithVL()

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to double value, contains the last "scale with VL" factor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.13 Methods for reading color measurment values

Functions

- 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 Detailed Description

5.13.2 Function Documentation

5.13.2.1 GOMDBTS2048_calculateColor()

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

If the calculation of the color values should not be active (setColorCalculation(false)), one can also initiate the calculation manually. This is not necessary if the calculation is active (setColorCalculation(true)) since the calculation is hereby performed automatically after the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.13.2.2 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 )
```

This method returns all computed color values based on a spectral measurement. The spectral measurement and the color calculation must have been activated before the measurement.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*UpperX*	pointer to a double value, contains “large” X
out	*UpperY*	pointer to a double value, contains “large” Y
out	*UpperZ*	pointer to a double value, contains “large” Z
out	*x*	pointer to a double value, contains x corresponding to the CIE1931 color space
out	*y*	pointer to a double value, contains y corresponding to the CIE1931 color space
out	*us*	pointer to a double value, contains u’ corresponding to the CIE1976 color space
out	*vs*	pointer to a double value, contains v’ corresponding to the CIE1976 color space
out	*CCT*	pointer to a double value, “Correlated Color Temperature”, the color temperature in [K]
out	*domWL*	pointer to a double value, contains the dominant wavelength

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

Example:

```
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);
```

5.13.2.3 GOMDBTS2048_getDeltaUV()

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

Returns the available delta uv value from the last measurement.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*uv*	Pointer to a double value, contains the determined delta uv value after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.13.2.4 GOMDBTS2048_getPurity()

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

Returns the color purity of the last measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, contains the determined purity value after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.13.2.5 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 )
```

Color Rendering Index refers to a photometric value describing the color rendering quality of light sources with the same correlated color temperature. Light with a color temperature of up to 5000 K that is emitted by a black body radiator with the corresponding color temperature is used as a reference for the rendering quality. Light with a color temperature greater than 5000 K is referenced using a daylight-like spectral distribution. For example, in order to calculate the color rendering index of a household bulb (which can be said to be a black body radiator), the spectrum of a black body radiator with a temperature of 2700 K is used as the reference whereas for a fluorescent lamp with 865 luminous color (865 for a color rendering index of more than 80, 865 for a color temperature of 6500 K), a daylight spectrum of the standard illuminant type D65 is used. By definition, the color rendering index is a special metamerism index. 14 test colors with a standard remission curve are defined for calculation of the color rendering index. The deviation of the secondary spectra between the reference and test spectrum is used as a measure for the 14 special color rendering indices. However, for calculation of the general color rendering index, Ra, only the first eight test colors are used. The 14 test colors are selected as per DIN 6169. This thus makes it possible to calculate the Ri color rendering index for the color i. An analytical value from the colors #1 to #8 is denoted by Ra. Since while defining the color rendering index in the 1930’s reference lamps with 100 were set with 50 (rather random), and since the color rendering index is not a percentual value, negative color rendering indices are also possible.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Parameters

`out`	*Ra*	Pointer to a double value, average value from R1 – R8
`out`	*R1*	Pointer to a double value, value of R1
`out`	*R2*	Pointer to a double value, value of R2
`out`	*R3*	Pointer to a double value, value of R3
`out`	*R4*	Pointer to a double value, value of von R4
`out`	*R5*	Pointer to a double value, value of R5
`out`	*R6*	Pointer to a double value, value of R6
`out`	*R7*	Pointer to a double value, value of R7
`out`	*R8*	Pointer to a double value, value of R8
`out`	*R9*	Pointer to a double value, value of R9
`out`	*R10*	Pointer to a double value, value of R10
`out`	*R11*	Pointer to a double value, value of R11
`out`	*R12*	Pointer to a double value, value of R12
`out`	*R13*	Pointer to a double value, value of R13
`out`	*R14*	Pointer to a double value, value of R14
`out`	*R15*	Pointer to a double value, value of R15

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14 Methods for trigger setting

Functions

- 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 Detailed Description

5.14.2 Function Documentation

5.14.2.1 GOMDBTS2048_setTriggerSource()

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

This method defines, if a measurement device should be triggered via external or internal trigger. The value „0“ defines, that the external trigger is active. The value „1“ defines, that the internal trigger is active.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Integer value, contains the determined trigger source: • 0: External trigger • 1: Internal trigger

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.2 GOMDBTS2048_getTriggerSource()

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

This method returns the current determined trigger source. That means, is the device triggerd by an extern or intern trigger signal.

- 0 -> external
- 1 -> internal

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to a integer value • 0: External trigger • 1: Internal trigger

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.3 GOMDBTS2048_setTriggerInternalLevels()

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

This Method defines the levels for the internal trigger. The first parameter contains the lightvalue, which should be used as trigger threshold. The second parameter contains the lightvalue which will be expected as maximum light value of the measurement. The light value always relates to the selected calibration entry. Triggered measurements should always be done in autoranging mode. So this method does not only set the trigger level but also switches off „autoranging mode“ and defines correct range, which is suitable for given maximum light value. The trigger threshold value (lightValueTrigger) mustn’t go below the limit of 1% of the maximum permissable light value (lightValueMax).

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*lightValueTrigger*	Integer value, which contains the light value, that will be used as trigger threshold.
in	*lightValueMax*	Integer value, which contains the maximum expected light value of the measurement.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.4 GOMDBTS2048_setTriggerMode()

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

This method specifies if a measurement device should react to edges or levels. In addition, the method “set↩ TriggerLevel” should be used to specify if the measurement device should switch upon detection of an edge (falling or rising) or a fixed level (low or high).

Parameters

in	*handle*	Integer value >0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value, contains the desired trigger mode: • 0: Level • 1: Edge

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.5 GOMDBTS2048_getTriggerMode()

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

This method determines the set trigger mode i.e. if the device should react to an edge or a fixed level at the trigger input.

Parameters

in	*handle*	Integer value >0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*mode*	Pointer to a integer value, contains the desired trigger mode • 1: Level • 1: Edge

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.6 GOMDBTS2048_setTriggerLevel()

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

this method specifies if the measurement device should react to a falling edge/ low level at the trigger input or to a rising edge/high level. This method should always be used together with the "setTriggerMode" method in specifying whether the device should react to the edges or level.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*level*	Double value, contains the desired level: • 0: Falling edge or low level • 1: Rising edge or high level

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.14.2.7 GOMDBTS2048_getTriggerLevel()

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

this method determines the setting for the trigger level i.e. if the device should react to the presence of a falling edge / low level or a rising edge / high level at the trigger input.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*level*	pointer to a double value, contains the desired trigger mode • 0: Falling / low • 1: Rising / high

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.14.2.8 GOMDBTS2048_setTriggerInput()

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

this method defines the trigger input that the device should react to.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*input*	integer value, contains the desired trigger input: • 1: Trigger input 1 • 2: Trigger input 2 • 3: Trigger input 3

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.14.2.9 GOMDBTS2048_getTriggerInput()

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

This method returns the set trigger input that the BTS2048 should react to.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*input*	pointer to an integer value, contains the selected trigger input: • 1: Trigger input 1 • 2: Trigger input 2 • 3: Trigger input 3

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.14.2.10 GOMDBTS2048_isMeasurementFinished()

```
int __stdcall GOMDBTS2048_isMeasurementFinished (
             int handle,
             bool * value )
```

this method checks whether a triggered measurement has already been completed. For a triggered measurement, this method has to return "true" before another method is called.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a Boolean value: • true: measurement is completed • false: measurement not yet completed

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.11 GOMDBTS2048_setTriggerTimeoutInMs()

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

This method specifies the timeout time for a triggered measurement. After the waiting time is over, the triggered measurement is cancelled.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value; the timeout time in milliseconds [ms].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.12 GOMDBTS2048_getTriggerTimeoutInMs()

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

Returns the previously set timeout time for triggered measurements.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value; the timeout time in milliseconds.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.13 GOMDBTS2048_setOut1LowDuringMeasurement()

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

This method specifies that the device output 1 pin of the measurement device is set to “low” directly before the integration time of of the spectral array starts. When the integration time of the measurement process is completed the output 1 value is set to "high". This makes it possible for an externally triggered device to detect the completion of the integration time of a measurement process.
Further processing (calculations, etc.) are not yet completed at this point.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean value: • false: output 1 not set to low during the measurement. No functionality of output 1 pin. • true: output 1 set to low during the measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.14 GOMDBTS2048_getOut1LowDuringMeasurement()

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

This method returns if the output 1 is set to "Low" during measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to a double value; contains the status after return: • false: Output 1 not set to Low, • true: Output 1 set to low during the measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.15 GOMDBTS2048_setTriggerDelay()

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

You can adjust the trigger delay with this method. The delay time [ms] is the time after a triggered measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*timeInMs*	Integer value, trigger delay in ms

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.14.2.16 GOMDBTS2048_getTriggerDelay()

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

Returns the delay time which is set in the device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to integer value, contains trigger delay in ms.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15 Methods for self absorbtion correction DUT (Device Under Test)

methods for self absorbtion correction with a device under test.

Functions

- 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 Detailed Description

5.15.2 Function Documentation

5.15.2.1 GOMDBTS2048_substitutionEnableCorrection()

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

This method enables and disables substitution correction of a measurement object (DUT = device under test). Substitution correction is an important step when performing measurements with an integrating sphere. The calibration values which are saved in the BTS2048 for measurement of the luminous flux are determined using an empty sphere. As soon as a measurement object is placed in the sphere, the properties of the sphere change. This change, which would result in a measurement error, must be compensated for. This is done through substitution correction.

For this method to be effective, substitution factors for the measurement object must be determined before the measurement. This is done using the “substitutionMeasurementWithoutDevice” and “substitutionMeasurement←↪ WithDevice” methods. These factors can be saved (“substitutionLoadFactors”) and later loaded (“substitution←↪ LoadFactors”). Initially, all factors are saved with “1.0”. This is the neutral factor that does not lead to correction.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*active*	Boolean value that determines if the substitution correction will be activated or not: • true: Substitution correction activated • false: Substitution correction deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.2 GOMDBTS2048_substitutionIsEnabledCorrection()

```
int __stdcall GOMDBTS2048_substitutionIsEnabledCorrection (
            int handle,
            bool * active )
```

Checks if substitution correction is enabled or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*active*	Pointer to a boolean value, contains information on whether the substitution correction is active or not: • true: Substitution correction is active • false: Substitution correction is not active

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.3 GOMDBTS2048_substitutionMeasurementWithoutTestDevice()

```
int __stdcall GOMDBTS2048_substitutionMeasurementWithoutTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This method should always be called first in order to perform a substitution measurement for a certain measurement object in an integrating sphere. The first step is always the measurement of the empty sphere. A substitution measurement can then be performed using “substitutionMeasurementWithTestDevice”.
For measurement of the empty sphere, the auxiliary lamp that was supplied with the sphere has to be switched

on. Let the auxiliary lamp burn in for the predefined time in order to have a stable signal. Try to determine the integration time such that a 54% to 95% saturation level is achieved. Small modulations may result in inaccurate substitution factors. Remove all the contents from the sphere and call this method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value, currently irrelevant!
out	*saturation*	Pointer to a double value, contains the modulation of the performed substitution measurement in [%].
out	*counts*	Array with double values, size 2048 elements, contains raw values (counts) of the spectral measurement.
out	*current*	Pointer to a double value, contains the raw value (current) of the integral sensor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.4 GOMDBTS2048_substitutionMeasurementWithTestDevice()

```
int __stdcall GOMDBTS2048_substitutionMeasurementWithTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This method should be called second to compute substitution factors. Switch on your auxiliary lamp and wait for the burn in time predefined by the manufacturer in order to obtain a stable signal.
While performing the measurement with the measurement object in the sphere, always use the same integration time as the one used with the empty sphere.
Repeat the measurement of the empty sphere using a different integration time if the dynamic range is not within 54% and 95%.
For certain measurement objects, it is not possible to attain the optimum dynamic range for both measurements (with an empty sphere and with the measurement object inside the sphere). In such cases, try to attain the best possible modulation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value, currently of no importance!
out	*saturation*	Pointer to a double value, contains the modulation of the substitution measurements in [%]
out	*counts*	Array with double values, size - 2048 elements, contains raw values (counts) of the spectral measurement
out	*current*	Pointer to a double value, contains raw value (current) of the integral sensor

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.5 GOMDBTS2048_substitutionSetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionSetIntegrationTimeInUs (
            int handle,
            int timeInus )
```

With this method it is possible to set the integration time in µs, which is used for the substitution measurement. Value range is 2 - 4000000 -> 2µs bis 4sec. If you have a device with cooling (cooling have to switch ON), it is valid to use a value up to 60000000 µs (60 sec.). If the integration time selected to high, the value is overloaded and not usable.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*timeInus*	Integer value, integration time in microseconds [µs].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.6 GOMDBTS2048_substitutionGetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionGetIntegrationTimeInUs (
            int handle,
            int * timeInus )
```

Returns the last selcted integration time in µs.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*timeInus*	Pointer to integer value, contains the integration time in microseconds [µs].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.7 GOMDBTS2048_substitutionSetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionSetDynamicTimeMode (
            int handle,
            bool value )
```

This method activates the dynamic adjustment of the integration time for the substitution.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean Value: • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.8 GOMDBTS2048_substitutionGetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionGetDynamicTimeMode (
            int handle,
            bool * value )
```

This method returns the dynamic time mode for the substitution.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to Boolean; containing the „time mode“: • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.9 GOMDBTS2048_substitutionSetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionSetHighResolutionMode (
            int handle,
            bool value )
```

This methode sets the high resolution mode for the substitution measurement. With high resolution mode activated, the spectrum is combined from different measurements with different integration times.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean Value: • false: high resolution deactiveted • true: high resolution activeted

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.10 GOMDBTS2048_substitutionGetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionGetHighResolutionMode (
            int handle,
            bool * value )
```

Returns the status of the high resolution mode for the substitution

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to Boolean: • false: high resolution deactiveted • true: high resolution activeted

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.11 GOMDBTS2048_substitutionSaveFactors()

```
int __stdcall GOMDBTS2048_substitutionSaveFactors (
            int handle,
            char * absoluteFileName )
```

Currently computed substitution factors are saved under the specified file name. The file name must be specified as a complete path.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*absoluteFileName*	String value, complete path to target file.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.12 GOMDBTS2048_substitutionLoadFactors()

```
int __stdcall GOMDBTS2048_substitutionLoadFactors (
            int handle,
            char * absoluteFileName )
```

Loads the previously measured and saved substitution factors from file into measurement device. These factors will be used when substitution correction is enabled by calling the “substitutionEnableCorrection” method.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*absoluteFileName*	Zero terminated string, contains the complete path to the file to be loaded.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.13 GOMDBTS2048_substitutionGetLoadedFilename()

```
int __stdcall GOMDBTS2048_substitutionGetLoadedFilename (
             int handle,
             char * absoluteFileName )
```

Returns the file name with the complete file path from which the current substitution factors should be loaded. If the substitution factors are not loaded from the file, an empty string is returned.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*absoluteFileName*	Zero terminated string, must be allocated 2048 bytes.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.14 GOMDBTS2048_substitutionGetSpectralFactor()

```
int __stdcall GOMDBTS2048_substitutionGetSpectralFactor (
             int handle,
             int pixelNumber,
             double * factor )
```

This method provides the current substitution factor for a specific pixel of the spectrometer. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGetPresetSpectralFactors method should be used.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*pixelNumber*	Integer value, contains the desired pixel number, value range: 0 - 2047.
`out`	*factor*	pointer to a double value, contains the substitution factor for the specified pixel

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.15.2.15 GOMDBTS2048_substitutionGetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all 2048 current substitution factors of the spectrometer. These factors are used in the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factors that are currently present (regardless of whether substitution correction is active or not) are determined, the substitutionGetPresetSpectralFactors method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to the first Value of a Double Array, containing the substitution factors of all pixel. Allocated with 2048 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.15.2.16 GOMDBTS2048_substitutionGetPresetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGetPresetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all the default substitution factors for the spectral sensor, regardless of whether the substitution correction is on or not. These factors only apply when the substitution correction is activated. The substitutionGet↩ SpectralFactor method should be used for the correction factors actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to the first value of a Double array, containing the substitution factors of all pixel. Allocated with 2048 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.15.2.17 GOMDBTS2048_substitutionGetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGetIntegralFactor (
            int handle,
            double * factor )
```

This method provides the current substitution factor for the integral sensor. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGetPresetIntegralFactor method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to a double value, contains the substitution factor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.15.2.18 GOMDBTS2048_substitutionGetPresetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGetPresetIntegralFactor (
            int handle,
            double * factor )
```

This method provides the default substitution factor for the integral sensor, regardless of whether the substitution correction is on or not. This factor is only used when the substitution correction is activated. The method substitutionGetIntegralFactor should be used for the correction factor actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to Double, containing hte substitutionfactor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.15.2.19 GOMDBTS2048_substitutionSetComment()

```
int __stdcall GOMDBTS2048_substitutionSetComment (
            int handle,
            char * comment )
```

This method sets a comment describing the current substitution more closely. It should be called before the current substitution correction is saved, since this comment is also stored.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*comment*	null-terminated string containing the comment; Maximum permissible length including terminator: 1024 bytes.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.20 GOMDBTS2048_substitutionGetComment()

```
int __stdcall GOMDBTS2048_substitutionGetComment (
             int handle,
             char * comment )
```

This method returns the comment set by the substitutionSetComment method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*comment*	null-terminated string, must be allocated with 1024 bytes; contains the comment

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.15.2.21 GOMDBTS2048_substitutionGetDateTime()

```
int __stdcall GOMDBTS2048_substitutionGetDateTime (
             int handle,
             int * day,
             int * month,
             int * year,
             int * hh,
             int * mm,
             int * ss )
```

This method returns the date and timethe current substitution corrction was stored.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*day*	Pointer to integer, returns the day of the date
out	*month*	Pointer to integer, returns the month of the date
out	*year*	Pointer to Integer, contains the year of the date
out	*hh*	Pointer to integer, contains the hours of the time after the jump
out	*mm*	Pointer to integer, contains the minutes of the time after the jump
out	*ss*	Pointer to integer, returns the seconds of the time

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16 Methods for self absorbtion correction geometrie

methods for self absorbtion correction with the sphere geometrie

Functions

- 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 Detailed Description

5.16.2 Function Documentation

5.16.2.1 GOMDBTS2048_substitutionGeoEnableCorrection()

```
int __stdcall GOMDBTS2048_substitutionGeoEnableCorrection (
             int handle,
             bool active )
```

This method enables and disables substitution correction of the geometry. Substitution correction is an important step when performing measurements with an integrating sphere. The calibration values which are saved in the BTS2048 for measurement of the luminous flux are determined using an empty sphere. As soon as the geometriy of the sphere is changed, the reflection properties of the sphere change. This change, which would result in a measurement error, must be compensated for. This is done through substitution correction.
For this method to be effective, substitution factors for the measurement object must be determined before the measurement. This is done using the “substitutionGeoMeasurementWithoutDevice” and “substitutionGeo←↩ MeasurementWithDevice” methods. These factors can be saved (“substitutionGeoLoadFactors”) and later loaded (“substitutionGeoLoadFactors”). Initially, all factors are saved with “1.0”. This is the neutral factor that does not lead to correction.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*active*	Boolean value that determines if the substitution correction will be activated or not: • true: Substitution correction activated • false: Substitution correction deactivated

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.2 GOMDBTS2048_substitutionGeoIsEnabledCorrection()

```
int __stdcall GOMDBTS2048_substitutionGeoIsEnabledCorrection (
            int handle,
            bool * active )
```

Checks if substitution correction of the geometrie is enabled or not.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*active*	Pointer to a boolean value, contains information on whether the substitution correction is active or not: • true: Substitution correction is active • false: Substitution correction is not active

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.3 GOMDBTS2048_substitutionGeoMeasurementWithoutTestDevice()

```
int __stdcall GOMDBTS2048_substitutionGeoMeasurementWithoutTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This method should always be called first in order to perform a substitution measurement for a certain measurement object in an integrating sphere. The first step is always the measurement of the empty sphere. A substitution measurement can then be performed using “substitutionMeasurementWithTestDevice”.
For measurement of the empty sphere, the auxiliary lamp that was supplied with the sphere has to be switched

on. Let the auxiliary lamp burn in for the predefined time in order to have a stable signal. Try to determine the integration time such that a 54% to 95% saturation level is achieved. Small modulations may result in inaccurate substitution factors. Remove all the contents from the sphere and call this method.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value, currently irrelevant!
out	*saturation*	Pointer to a double value, contains the modulation of the performed substitution measurement in [%].
out	*counts*	Array with double values, size 2048 elements, contains raw values (counts) of the spectral measurement.
out	*current*	Pointer to a double value, contains the raw value (current) of the integral sensor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.4 GOMDBTS2048_substitutionGeoMeasurementWithTestDevice()

```
int __stdcall GOMDBTS2048_substitutionGeoMeasurementWithTestDevice (
            int handle,
            bool isExternalSphere,
            double * saturation,
            double * counts,
            double * current )
```

This method should be called second to compute substitution factors. Switch on your auxiliary lamp and wait for the burn in time predefined by the manufacturer in order to obtain a stable signal.
While performing the measurement with the measurement object in the sphere, always use the same integration time as the one used with the empty sphere.
Repeat the measurement of the empty sphere using a different integration time if the dynamic range is not within 54% and 95%.
For certain measurement objects, it is not possible to attain the optimum dynamic range for both measurements (with an empty sphere and with the measurement object inside the sphere). In such cases, try to attain the best possible modulation.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*isExternalSphere*	Boolean value, currently of no importance!
out	*saturation*	Pointer to a double value, contains the modulation of the substitution measurements in [%]
out	*counts*	Array with double values, size - 2048 elements, contains raw values (counts) of the spectral measurement
out	*current*	Pointer to a double value, contains raw value (current) of the integral sensor

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.5 GOMDBTS2048_substitutionGeoSetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionGeoSetIntegrationTimeInUs (
            int handle,
            int timeInus )
```

With this method it is possible to set the integration time in µs, which is used for the substitution measurement. Value range is 2 - 4000000 -> 2µs bis 4sec. If you have a device with cooling (cooling have to switch ON), it is valid to use a value up to 60000000 µs (60 sec.). If the integration time selected to high, the value is overloaded and not usable.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*timeInus*	Integer value, integration time in microseconds [µs].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.6 GOMDBTS2048_substitutionGeoGetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_substitutionGeoGetIntegrationTimeInUs (
            int handle,
            int * timeInus )
```

Returns the last selcted integration time in µs.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*timeInus*	Pointer to integer value, contains the integration time in microseconds [µs].

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.7 GOMDBTS2048_substitutionGeoSetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionGeoSetDynamicTimeMode (
            int handle,
            bool value )
```

This method activates the dynamic adjustment of the integration time for the substitution.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean Value: • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.16.2.8 GOMDBTS2048_substitutionGeoGetDynamicTimeMode()

```
int __stdcall GOMDBTS2048_substitutionGeoGetDynamicTimeMode (
            int handle,
            bool * value )
```

This method returns the dynamic time mode for the substitution.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to Boolean; containing the „time mode": • true: dynamic mode active • false: dynamic mode not active

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.16.2.9 GOMDBTS2048_substitutionGeoSetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionGeoSetHighResolutionMode (
            int handle,
            bool value )
```

This methode sets the high resolution mode for the substitution measurement. With high resolution mode activated, the spectrum is combined from different measurements with different integration times.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Boolean Value: • false: high resolution deactiveted • true: high resolution activeted

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.10 GOMDBTS2048_substitutionGeoGetHighResolutionMode()

```
int __stdcall GOMDBTS2048_substitutionGeoGetHighResolutionMode (
            int handle,
            bool * value )
```

Returns the status of the high resolution mode for the substitution

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to Boolean: • false: high resolution deactiveted • true: high resolution activeted

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.11 GOMDBTS2048_substitutionGeoSaveFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoSaveFactors (
            int handle,
            char * absoluteFileName )
```

Currently computed substitution factors are saved under the specified file name. The file name must be specified as a complete path.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*absoluteFileName*	String value, complete path to target file.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.12 GOMDBTS2048_substitutionGeoLoadFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoLoadFactors (
            int handle,
            char * absoluteFileName )
```

Loads the previously measured and saved substitution factors from file into measurement device. These factors will be used when substitution correction is enabled by calling the “substitutionEnableCorrection” method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*absoluteFileName*	Zero terminated string, contains the complete path to the file to be loaded.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.13 GOMDBTS2048_substitutionGeoGetLoadedFilename()

```
int __stdcall GOMDBTS2048_substitutionGeoGetLoadedFilename (
             int handle,
             char * absoluteFileName )
```

Returns the file name with the complete file path from which the current substitution factors should be loaded. If the substitution factors are not loaded from the file, an empty string is returned.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*absoluteFileName*	Zero terminated string, must be allocated 2048 bytes.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.14 GOMDBTS2048_substitutionGeoGetSpectralFactor()

```
int __stdcall GOMDBTS2048_substitutionGeoGetSpectralFactor (
             int handle,
             int pixelNumber,
             double * factor )
```

This method provides the current substitution factor for a specific pixel of the spectrometer. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGeoGetPresetSpectralFactors method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*pixelNumber*	Integer value, contains the desired pixel number, value range: 0 - 2047.
out	*factor*	pointer to a double value, contains the substitution factor for the specified pixel

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.15 GOMDBTS2048_substitutionGeoGetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoGetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all 2048 current substitution factors for a specific pixel of the spectrometer. These factors are used in the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factors that are currently present (regardless of whether substitution correction is active or not) are determined, the substitutionGeoGetPresetSpectralFactors method should be used.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to the first Value of a Double Array, containing the substitution factors of all pixel. Allocated with 2048 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.16 GOMDBTS2048_substitutionGeoGetPresetSpectralFactors()

```
int __stdcall GOMDBTS2048_substitutionGeoGetPresetSpectralFactors (
            int handle,
            double * factor )
```

This method provides all the default substitution factors for the spectral sensor, regardless of whether the substitution correction is on or not. These factors only apply when the substitution correction is activated. The substitution↩ GeoGetSpectralFactor method should be used for the correction factors actually used during the measurement.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to the first value of a Double array, containing the substitution factors of all pixel. Allocated with 2048 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.17 GOMDBTS2048_substitutionGeoGetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGeoGetIntegralFactor (
            int handle,
            double * factor )
```

This method provides the current substitution factor for the integral sensor. This factor is used for the current measurement. If the substitution correction is switched off, the value 1.0 is always returned here. If the default substitution factor that is currently present (regardless of whether substitution correction is active or not) is to be determined, the substitutionGeoGetPresetIntegralFactor method should be used.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to a double value, contains the substitution factor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.18 GOMDBTS2048_substitutionGeoGetPresetIntegralFactor()

```
int __stdcall GOMDBTS2048_substitutionGeoGetPresetIntegralFactor (
            int handle,
            double * factor )
```

This method provides the default substitution factor for the integral sensor, regardless of whether the substitution correction is on or not. This factor is only used when the substitution correction is activated. The method substitutionGeoGetIntegralFactor should be used for the correction factor actually used during the measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*factor*	Pointer to Double, containing hte substitutionfactor.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.19 GOMDBTS2048_substitutionGeoSetComment()

```
int __stdcall GOMDBTS2048_substitutionGeoSetComment (
            int handle,
            char * comment )
```

This method sets a comment describing the current substitution more closely. It should be called before the current substitution correction is saved, since this comment is also stored.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*comment*	null-terminated string containing the comment; Maximum permissible length including terminator: 1024 bytes.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.20 GOMDBTS2048_substitutionGeoGetComment()

```
int __stdcall GOMDBTS2048_substitutionGeoGetComment (
            int handle,
            char * comment )
```

This method returns the comment set by the substitutionGeoSetComment method.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*comment*	null-terminated string, must be allocated with 1024 bytes; contains the comment

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.16.2.21 GOMDBTS2048_substitutionGeoGetDateTime()

```
int __stdcall GOMDBTS2048_substitutionGeoGetDateTime (
            int handle,
            int * day,
            int * month,
            int * year,
            int * hh,
            int * mm,
            int * ss )
```

This method returns the date and timethe current substitution corrction was stored.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*day*	Pointer to integer, returns the day of the date
out	*month*	Pointer to integer, returns the month of the date
out	*year*	Pointer to Integer, contains the year of the date
out	*hh*	Pointer to integer, contains the hours of the time after the jump
out	*mm*	Pointer to integer, contains the minutes of the time after the jump
out	*ss*	Pointer to integer, returns the seconds of the time

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17 Base calibartion methods

Functions

- 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 Detailed Description

5.17.2 Function Documentation

5.17.2.1 GOMDBTS2048_calibLoadFromDevice()

```
int __stdcall GOMDBTS2048_calibLoadFromDevice (
            int handle )
```

this method loads the calibration data of the currently selected calibration configuration ("setCalibrationEntry↩ Number") from the measurement device in the buffer memory. The data can then be read from the buffer memory using the "Get" methods. If one wants to change certain values in a calibration configuration, it is recommended that the existing configuration be loaded first, the desired changes be made and then saved. Loading of calibration data can last up to one minute.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.17.2.2 GOMDBTS2048_calibSaveToDevice()

```
int __stdcall GOMDBTS2048_calibSaveToDevice (
            int handle,
            int configNumber )
```

The currently defined calibration data is saved in the measurement device. Before saving, it is checked whether all the required parameters have been set. If not, an error code is returned.
The “user-specific” memory is defined in numbers 15 – 23. The positions 0 – 14 are reserved for factory calibrations by Gigahertz-Optik.
The use of this method is rejected if the given password for calibrations is not accepted. The process of saving can take up to one minute.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*configNumber*	Integer value; configuration number under which this calibration is saved in the device, 0 – 14 are reserved for factory calibrations, 15 – 23 are freely available

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.3 GOMDBTS2048_calibSetCalibLampFileName()

```
int __stdcall GOMDBTS2048_calibSetCalibLampFileName (
            int handle,
            char * filename )
```

Loads calibration data from an external file and uses this data to calibrate. If this method is used, the "calib←AzSetCalibLamp" method can not be used. If the calibration factors are to be determined using the calibration measurement methods, this method must be used instead of "calibAzSetCalibLamp". \ n The calibration lamp data must be available in unit [W]. The format for calibration files is as follows:
Line: (optional) Comment line, marked by "//" or ";" at the beginning of the line \ n The following lines: in each line an entry (wavelength and associated lamp value separated by tab) \ N Example: \ n // comment line \ n 250 124.365 \ n 255 166.447 \ n 260 215,786 \ n 265 278,089 \ n ...

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*filename*	nullterminated string; contains complete path to the lamp-file

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.4 GOMDBTS2048_calibGetCalibLampFileName()

```
int __stdcall GOMDBTS2048_calibGetCalibLampFileName (
            int handle,
            char * filename )
```

Returns the previously defined calibration-filename with the complete path. If the calibration-lamp data are set using the method "calibAzSetCalibLamp", then the file name is empty.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*filename*	null-terminated string; Contains the complete path to the lamp file, the string must be allocated with 2048 bytes.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.5 GOMDBTS2048_calibMeasureSpectral()

```
int __stdcall GOMDBTS2048_calibMeasureSpectral (
            int handle,
            double * saturation,
            double * countsDark,
            double * countsSignal,
            double * calFactors )
```

With this method the spectral calibration factors can be determined by a calibration measurement. Before this, the calibration lamp must have been set with "calibSetCalibLampFileName".

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*saturation*	pointer to double value; Contains the control of the spectral unit during the calibration measurement
out	*countsDark*	Pointer on the first element of a double array, contains the dark signal of the spectral unit during the calibration measurement; Memory needs to be allocated for 2048 double values
out	*countsSignal*	Pointer on first element of a double array, contains the useful signal of the spectral unit during the calibration measurement; Memory needs to be allocated for 2048 double values
out	*calFactors*	Pointer on first element of a double array, contains the determined calibration factors of the spectral unit at the Calibration; Memory needs to be allocated for 2048 double values

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.6 GOMDBTS2048_calibMeasureIntegral()

```
int __stdcall GOMDBTS2048_calibMeasureIntegral (
            int handle,
            double * currentDark,
            double * currentSignal,
            double * calFactor )
```

With this method the spectral calibration factor can be determined by a calibration measurement. Before this, the calibration lamp must have been set with "calibSetCalibLampFileName".

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*currentDark*	pointer to double value, contains the dark current of the integral unit during the calibration measurement
`out`	*currentSignal*	pointer to double value, contains the useful current of the integral unit during the calibration measurement
`out`	*calFactor*	pointer to double value, contains the determined calibration factor of the integral unit during the calibration measurement

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.7 GOMDBTS2048_calibSetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_calibSetIntegrationTimeInUs (
            int handle,
            int value )
```

This method defines the integration time to be used in the spectral calibration measurement. If saturation is too low (should be at least 54%), the integration time should be greater.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*value*	double value, contains the integration time in microseconds

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.8 GOMDBTS2048_calibGetIntegrationTimeInUs()

```
int __stdcall GOMDBTS2048_calibGetIntegrationTimeInUs (
            int handle,
            int * value )
```

This method defines the integration time to be used in the spectral calibration measurement. If saturation is too low (should be at least 54%), the integration time should be greater.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*value*	pointer to double value, contains the integration time in microseconds

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.9 GOMDBTS2048_calibSetCalibrationName()

```
int __stdcall GOMDBTS2048_calibSetCalibrationName (
            int handle,
            char * name )
```

This method defines the name of the calibration configuration; this is displayed in the configuration window. The name should be unambiguous so as not to cause confusion. Factory calibrations are always named as follows: “NAME (UNIT)”, whereby NAME can be any random name and UNIT the actual unit of the integral measurement unit.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*name*	Zero terminated string, max. length: 31 characters plus zero terminator

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.10 GOMDBTS2048_calibGetCalibrationName()

```
int __stdcall GOMDBTS2048_calibGetCalibrationName (
            int handle,
            char * name )
```

This method returns the previously defined calibration name.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*name*	Zero terminated string; contains the configuration name after return, minimum size: 32 bytes

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

Example:

```
int handle;
GOMDBTS2048_getHandle(NULL, &handle);
char value[32];
GOMDBTS2048_calibLoadFromDevice(handle);
GOMDBTS2048_calibGetCalibrationName(handle, value);
//do anything with values
GOMDBTS2048_releaseHandle(handle);
```

5.17.2.11 GOMDBTS2048_calibSetCalibMode()

```
int __stdcall GOMDBTS2048_calibSetCalibMode (
            int handle,
            int value )
```

This method defines the calibration mode. The default value is 0 and corresponds to the standard calibration method. \ n Mode 0: Standard calibration method (default) \ n Mode 1: Resacuation. Instead of re-setting all calibration entries, all spectral entries are scaled with a factor so that the measured radiometric value matches that of the lamp file. \ n Mode 2: Calibration with several calibration lamps. With this Mosud several lamps with different WL range can be used. For the more detailed procedure, read the "Calibration with Multiple Calibration Lamps" in the introduction to this chapter.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	integer value: • 0: Normal calibration mode • 1: Rescaling instead of recalibration • 2: Calibration with several lamps

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.12 GOMDBTS2048_calibGetCalibMode()

```
int __stdcall GOMDBTS2048_calibGetCalibMode (
            int handle,
            int * value )
```

This method returns the calibration mode.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to integer value, contains the calibration mode after return. • 0: Normal calibration mode • 1: Rescaling instead of recalibration • 2: Calibration with several lamps

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.13 GOMDBTS2048_calibSetHighResolutionMode()

```
int __stdcall GOMDBTS2048_calibSetHighResolutionMode (
            int handle,
            int value )
```

This method switches the high-resolution measurement mode on and off for the calibMeasureSpectral () function. The spectrum is composed of several measurements with different integration times. The transfer parameter specifies the dynamic range (number of measurements). At 0, the mode is deactivated.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	integer value: • 0: High-resolution measurement • Otherwise($>$0): dynamic range (number of measurements with different integration time)

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.14 GOMDBTS2048_calibGetHighResolutionMode()

```
int __stdcall GOMDBTS2048_calibGetHighResolutionMode (
            int handle,
            int * value )
```

This method returns the status of the high-resolution calibration measurement.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to integer value: • 0: High-resolution measurement • Otherwise($>$0): dynamic range (number of measurements with different integration time)

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.17.2.15 GOMDBTS2048_calibCalculateSpectralCalibrationFactors()

```
int __stdcall GOMDBTS2048_calibCalculateSpectralCalibrationFactors (
            int handle,
            double * calFactors )
```

This is only used in calibration mode 2. After all calibration lamps have been measured, this function must be called once to combine all spectral measurements and to calculate the calibration factors.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*calFactors*	pointer on the first element of a double array, contains the determined calibration factors of the spectral unit during the calibration measurement; memory for 2048 double values must be allocated.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18 Manual calibration methods

Functions

- 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 Detailed Description

5.18.2 Function Documentation

5.18.2.1 GOMDBTS2048_calibNew()

```
int __stdcall GOMDBTS2048_calibNew (
            int handle )
```

Deletes the buffer memory i.e. releases the calibration's intermediate memory. This method must not be forcibly called since if the buffer memory is unavailable, it is automatically created by the "Set" methods. One only needs to use this method to confirm that previously saved calibration values have been deleted from the buffer memory.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.18.2.2 GOMDBTS2048_calibTristimulusGetXYZ()

```
int __stdcall GOMDBTS2048_calibTristimulusGetXYZ (
            int handle,
            double * valuesX,
            double * valuesY,
            double * valuesZ )
```

This method returns the X, Y, Z tristimulus values loaded using the method “calibTristimulusGetXYZ”.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*valuesX*	Pointer to the first element of a double array with a size of 500, contains the previously defined tristimulus values for X from 350nm to 849nm in 1nm steps after return -> exactly 500 values
out	*valuesY*	pointer to the first element of a double array with a size of 500, contains the previously defined tristimulus values for Y from 350nm to 849nm in 1nm steps after return -> exactly 500 values
out	*valuesZ*	pointer to the first element of a double array with a size of 500, contains the previously defined tristimulus values for Z from 350nm to 849nm in 1nm steps after return -> exactly 500 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.3 GOMDBTS2048_calibAzSetCalibLamp()

```
int __stdcall GOMDBTS2048_calibAzSetCalibLamp (
            int handle,
            double * values,
            char * name )
```

this method is used to save the spectrum of the calibration lamp. The relative spectral data is sufficient. A name can be given to the calibration lamp. The calibration lamp spectrum is required for the a(z) correction and is specifically saved for each measurement device and must therefore be set for each calibration configuration. The spectrum must be within a 350nm to 849nm range with a 1nm step size. Exactly 500 values are hereby required.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*values*	Double array, contains the calibration lamp spectrum from 350nm to 849nm in 1nm steps -> exactly 500 values
in	*name*	Zero terminated string, name of the calibration lamp (max. 31 characters plus zero terminator)

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.4 GOMDBTS2048_calibAzGetCalibLamp()

```
int __stdcall GOMDBTS2048_calibAzGetCalibLamp (
             int handle,
             double * values,
             char * name )
```

This method returns the defined calibration lamp spectrum and name of the calibration lamp.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*values*	Pointer to the first element of a double array with a size of 500, contains the defined calibration lamp spectrum from 350nm to 849nm in 1nm steps after return-$>$ exactly 500 values
out	*name*	Zero terminated string; contains the name of the calibration lamp after return, minimum size: 32 bytes

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.5 GOMDBTS2048_calibAzSetTransmissionFileActual()

```
int __stdcall GOMDBTS2048_calibAzSetTransmissionFileActual (
             int handle,
             char * absoluteFileName )
```

With this methode you can set the transmission of an ulbricht sphere. The values are stored in a text file with following format:
ANSI textfile
1. row: (optional) comment, with „//“ or „;“ at the beginning
following rows: one entry per row (wavelength and transmission seperated by a tab stop)
Exapmle:
//comment
250 124,365
255 166,447
260 215,786
265 278,089
. . .

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*absoluteFileName*	zero terminated string; file name including the absolute filename of the transmission file.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.6 GOMDBTS2048_calibAzSetWeightingFunctionActual()

```
int __stdcall GOMDBTS2048_calibAzSetWeightingFunctionActual (
            int handle,
            double * values )
```

This method sets the weighting function of the integral sensor e.g., actual photometric curve of the integral sensor. Pre-calculation and consideration of transmission curves might be necessary. The individual values range from 350nm to 849nm in 1nm steps. Exactly 500 values are required.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*values*	Double array, contains the weighting function of the integral sensor with regards to possible transmission curves ranging from 350nm to 849nm in 1nm steps -> exactly 500 values

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.7 GOMDBTS2048_calibAzGetWeightingFunctionActual()

```
int __stdcall GOMDBTS2048_calibAzGetWeightingFunctionActual (
            int handle,
            double * values )
```

This method returns the previously defined weighting function of the integral sensor.

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*values*	previously defined weighting function of the integral sensor along with regards to possible transmission curves from 350nm to 849nm in 1nm steps -> exactly 500 values

Returns

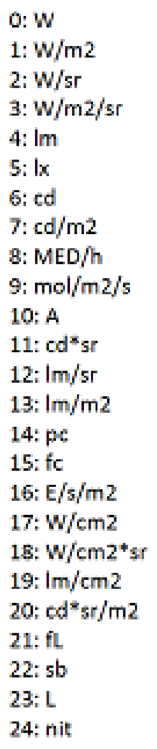
Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.8 GOMDBTS2048_calibSetCalibrationFactorsSpectral()

```
int __stdcall GOMDBTS2048_calibSetCalibrationFactorsSpectral (
            int handle,
            double * values )
```

This method is used to define calibration factors e.g., spectral unit. The spectral unit has 2048 pixels. A double array with 2048 calibration factors therefore has to be passed on (per calibration factor). The factors are required per pixel in: absolute value/use counts∗integration time/substitution factor.

- Units of the quantities:
- absolute value see table:



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

- use counts in cts
- integration time in seconds
- substitution factor, unitless

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*values*	Double array, contains the calibration factor for spectral measurement, exact 2048 values, each pixel one calibration factor

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.9 GOMDBTS2048_calibGetCalibrationFactorsSpectral()

```
int __stdcall GOMDBTS2048_calibGetCalibrationFactorsSpectral (
            int handle,
            double * values )
```

This method returns the previously defined calibration factors of the spectral unit.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*values*	values: pointer to the first element of a double array with a size of 2048, contains the previously defined calibration factors for all pixels after return -$>$ exactly 2048 values

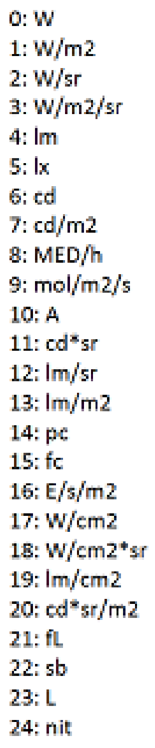
Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.10 GOMDBTS2048_calibSetUnitSpectral()

```
int __stdcall GOMDBTS2048_calibSetUnitSpectral (
            int handle,
            int value )
```

This method defines the calibration lamp unit for the spectral measurement e.g., W = 0. The following SI units table applies:


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

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*value*	Integer value, contains the unit number for the spectral measurement unit

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.11 GOMDBTS2048_calibGetUnitSpectral()

```
int __stdcall GOMDBTS2048_calibGetUnitSpectral (
            int handle,
            int * value )
```

This method returns the previously defined unit for the spectral measurement unit as per the units table.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to an integer value, contains the previously defined unit number as per the units table after return.

Returns

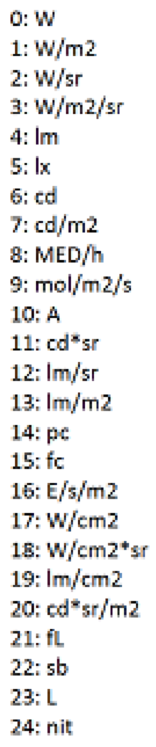
Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.12 GOMDBTS2048_calibSetCalibrationFactorIntegral()

```
int __stdcall GOMDBTS2048_calibSetCalibrationFactorIntegral (
            int handle,
            double value )
```

Description: this method is used to define the calibration factor of the integral unit. This is required in: Absolute value/measured current value/substitution factor. Units of the quantities:

- absolute value see table:



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

- measured current value in A
- substitutions factor, unitless

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	double value, contains the calibration factor for the integral unit

Returns

Integer values; see return value table (warnings and errors) for values unequal to "0".

5.18.2.13 GOMDBTS2048_calibGetCalibrationFactorIntegral()

```
int __stdcall GOMDBTS2048_calibGetCalibrationFactorIntegral (
            int handle,
            double * value )
```

This method returns the previously defined calibration factor of the integral unit.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a double value, contains the previously defined calibration factor for the integral unit after return.

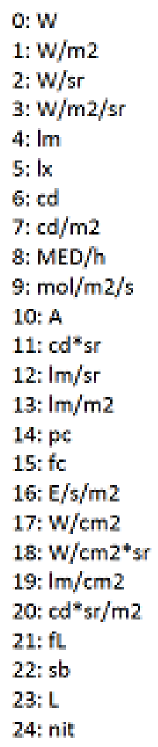
Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.14 GOMDBTS2048_calibSetUnitIntegral()

```
int __stdcall GOMDBTS2048_calibSetUnitIntegral (
             int handle,
             int value )
```

this method is used to define the SI unit with which the detector is calibrated e.g., lm = 4. The following SI units table applies:


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

Parameters

in	*handle*	Integer value > 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	integer value, contains the unit number for the integral unit.

Returns

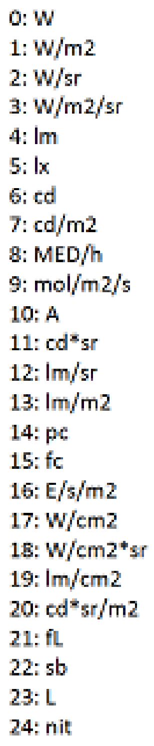
Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.15 GOMDBTS2048_calibGetUnitIntegral()

```
int __stdcall GOMDBTS2048_calibGetUnitIntegral (
             int handle,
             int * value )
```

This method returns the previously defined SI unit for the integral measurement unit as per the units table.

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`out`	*value*	Pointer to an integer value, contains the previously defined units number as per the units table after return Unit tabele:  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

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.16 GOMDBTS2048_calibSetFilterAssignment()

```
int __stdcall GOMDBTS2048_calibSetFilterAssignment (
            int handle,
            int value )
```

The BTS2048 has a filter wheel with 4 filter positions (open, OD1, OD2, closed). Each calibration always requires an associated filter position that is always valid for the calibration. This filter wheel position is always used for measurements even if another filter has been selected using “setFilterPosition” after selection of a calibration entry. The assignment of the filter wheel position can be done using this method. The following is a list of the filter wheel positions and their corresponding filters:

- 0: Closed
- 1: OD2
- 2: OD1
- 3: Open

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*value*	Integer value, contains the filter wheel position, possible values 0 - 3.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.17 GOMDBTS2048_calibGetFilterAssignment()

```
int __stdcall GOMDBTS2048_calibGetFilterAssignment (
             int handle,
             int * value )
```

This method returns the filter wheel position assigned to the particular calibration configuration.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	pointer to an integer value, contains the assigned filter wheel position after return • 0: Closed • 1: OD2 • 2: OD1 • 3: Open

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.18 GOMDBTS2048_calibSetExternalSphere()

```
int __stdcall GOMDBTS2048_calibSetExternalSphere (
             int handle,
             bool value )
```

this method defines whether or not the calibration is valid for the measurement setup with an external integrating sphere. If this parameter is not explicitly defined or if it had been previously defined, “false” (no sphere) is returned by default.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Parameters

in	*value*	Boolean value: • true: Calibration for the measurement setup with integrating sphere • false: Measurement setup with no integrating sphere

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.19 GOMDBTS2048_calibGetExternalSphere()

```
int __stdcall GOMDBTS2048_calibGetExternalSphere (
            int handle,
            bool * value )
```

This method returns information on whether or not the calibration is defined for a measurement setup with an integrating sphere.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*value*	Pointer to a Boolean value, after return, it contains information on an external integration sphere: • true: Measurement setup with integrating sphere • false: Measurement setup with no integrating sphere

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.18.2.20 GOMDBTS2048_calibTristimulusSetXYZ()

```
int __stdcall GOMDBTS2048_calibTristimulusSetXYZ (
            int handle,
            double * valuesX,
            double * valuesY,
            double * valuesZ )
```

This method is used to set the tristimulus weighting functions to the device. These are used to calculate color values X, Y and Z. curves ranging from 350nm to 849nm in 1nm steps -$>$ exactly 500 values

Parameters

`in`	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
`in`	*valuesX*	Pointer to Double array, contains the weighting function(500 values) for color calculation of X value
`in`	*valuesY*	Pointer to Double array, contains the weighting function(500 values) for color calculation of Y value
`in`	*valuesZ*	Pointer to Double array, contains the weighting function(500 values) for color calculation of Z value

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.19 Wavelength calibration methods

Functions

- 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 Detailed Description

5.19.2 Function Documentation

5.19.2.1 GOMDBTS2048_calibSetWavelengthMapping()

```
int __stdcall GOMDBTS2048_calibSetWavelengthMapping (
            int handle,
            double ∗ values )
```

With this method, the pixel wavelength allocation can be changed. Caution: The wave length assignment is temporarily stored in the DLL using this method. The method calibWavelengthSaveMapping must be called to save it in the device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*values*	double array, which contains pixels to wavelength allocations for all 2048 pixels of the BTS2048 in ascending order. The first value of the array corresponds to the first pixel, the last (2048th) value of the array corresponds to the last pixel.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.19.2.2 GOMDBTS2048_calibGetWavelengthMapping()

```
int __stdcall GOMDBTS2048_calibGetWavelengthMapping (
            int handle,
            double ∗ values )
```

This method provides the current pixel-wavelength allocation.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*values*	pointer to the first element of a double array of size 2048, contains the pixel-wavelength allocation after return.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.19.2.3 GOMDBTS2048_calibWavelengthMeasureLamp()

```
int __stdcall GOMDBTS2048_calibWavelengthMeasureLamp (
            int handle,
            int lampnumber )
```

This method is used to measure a specific calibration lamp. Integration time, overload and other factors are determined automatically. For successful wavelength calibration, each lamp must be measured once. The assignment of the lamp numbers is as follows:

- lampnumber 0: HgAr-VL-lamp
- lampnumber 1: Ne-VL-lamp
- lampnumber 2: Kr-VL-lamp

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
in	*lampnumber*	integer lampnumber (look description)

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.19.2.4 GOMDBTS2048_calibWavelengthCalculateMapping()

```
int __stdcall GOMDBTS2048_calibWavelengthCalculateMapping (
            int handle,
            double * mapping )
```

After all lamps have been measured, this method can be used to calculate the wavelength pixel allocation. Make sure that the file "calibWavelengthBTS2048.gdf" is found by the DLL. The file must either be located in the folder itself or under "Documents \ Gigahertz-Optik \ datacalib". Calculation and allocation are then performed automatically. This is temporarily stored in the DLL. The method calibWavelengthSaveMapping must be called to save it in the device.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.
out	*mapping*	pointer to the first element of a double array of size 2048, contains the pixel-wavelength allocation after return

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.

5.19.2.5 GOMDBTS2048_calibWavelengthSaveMapping()

```
int __stdcall GOMDBTS2048_calibWavelengthSaveMapping (
            int handle )
```

This method saves the temporarily stored wavelength allocation in the BTS2048. This overwrites the old wavelength allocation. Note that this can not be recovered. After changing the wavelength allocation, all calibrations stored in the BTS2048 are invalid and must be carried out again.

Parameters

in	*handle*	Integer value $>$ 0 for unique identification of the instantiated BTS2048; this value is returned by the getHandle method.

Returns

Integer values; see return value table (warnings and errors) for values unequal to “0”.