

NKTPDLL Reference manual

Generated by Doxygen 1.9.7

| | |
|---|----------|
| 1 Deprecated List | 1 |
| 2 Data Structure Index | 3 |
| 2.1 Data Structures | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Data Structure Documentation | 7 |
| 4.1 lvDeviceStatusStruct Struct Reference | 7 |
| 4.1.1 Detailed Description | 7 |
| 4.1.2 Field Documentation | 7 |
| 4.1.2.1 portname | 7 |
| 4.1.2.2 devId | 8 |
| 4.1.2.3 status | 8 |
| 4.1.2.4 devDataLen | 8 |
| 4.1.2.5 devData | 8 |
| 4.2 lvPortStatusStruct Struct Reference | 8 |
| 4.2.1 Detailed Description | 8 |
| 4.2.2 Field Documentation | 9 |
| 4.2.2.1 portname | 9 |
| 4.2.2.2 status | 9 |
| 4.2.2.3 curScanAdr | 9 |
| 4.2.2.4 maxScanAdr | 9 |
| 4.2.2.5 foundType | 9 |
| 4.3 lvRegisterStatusStruct Struct Reference | 9 |
| 4.3.1 Detailed Description | 10 |
| 4.3.2 Field Documentation | 10 |
| 4.3.2.1 portname | 10 |
| 4.3.2.2 devId | 10 |
| 4.3.2.3 regId | 10 |
| 4.3.2.4 status | 10 |
| 4.3.2.5 regType | 11 |
| 4.3.2.6 regDataLen | 11 |
| 4.3.2.7 regData | 11 |
| 4.4 tDateTimeStruct Struct Reference | 11 |
| 4.4.1 Detailed Description | 11 |
| 4.4.2 Field Documentation | 12 |
| 4.4.2.1 Sec | 12 |
| 4.4.2.2 Min | 12 |
| 4.4.2.3 Hour | 12 |
| 4.4.2.4 Day | 12 |
| 4.4.2.5 Month | 12 |

| | |
|--------------------------------------|-----------|
| 4.4.2.6 Year | 12 |
| 4.5 tParamSetStruct Struct Reference | 12 |
| 4.5.1 Detailed Description | 13 |
| 4.5.2 Field Documentation | 13 |
| 4.5.2.1 Unit | 13 |
| 4.5.2.2 ErrorHandler | 13 |
| 4.5.2.3 StartVal | 14 |
| 4.5.2.4 FactoryVal | 14 |
| 4.5.2.5 ULimit | 14 |
| 4.5.2.6 LLimit | 14 |
| 4.5.2.7 Numerator | 14 |
| 4.5.2.8 Denominator | 14 |
| 4.5.2.9 Offset | 14 |
| 5 File Documentation | 15 |
| 5.1 NKTPDLL.h File Reference | 15 |
| 5.1.1 Detailed Description | 26 |
| 5.1.2 Macro Definition Documentation | 26 |
| 5.1.2.1 NKTPDLL_EXPORT | 26 |
| 5.1.3 Typedef Documentation | 26 |
| 5.1.3.1 PortResultTypes | 26 |
| 5.1.3.2 P2PPortResultTypes | 26 |
| 5.1.3.3 DeviceResultTypes | 27 |
| 5.1.3.4 DeviceModeTypes | 27 |
| 5.1.3.5 RegisterResultTypes | 27 |
| 5.1.3.6 RegisterDataTypes | 27 |
| 5.1.3.7 RegisterPriorityTypes | 27 |
| 5.1.3.8 PortStatusTypes | 27 |
| 5.1.3.9 DeviceStatusTypes | 27 |
| 5.1.3.10 RegisterStatusTypes | 27 |
| 5.1.3.11 DateTimeType | 27 |
| 5.1.3.12 ParamSetUnitTypes | 27 |
| 5.1.3.13 ParameterSetType | 28 |
| 5.1.3.14 GetAllPortsFuncPtr | 28 |
| 5.1.3.15 GetOpenPortsFuncPtr | 28 |
| 5.1.3.16 PointToPointPortAddFuncPtr | 28 |
| 5.1.3.17 PointToPointPortGetFuncPtr | 28 |
| 5.1.3.18 PointToPointPortDelFuncPtr | 28 |
| 5.1.3.19 OpenPortsFuncPtr | 28 |
| 5.1.3.20 ClosePortsFuncPtr | 28 |
| 5.1.3.21 SetLegacyBusScanningFuncPtr | 29 |
| 5.1.3.22 GetLegacyBusScanningFuncPtr | 29 |

| | |
|---|----|
| 5.1.3.23 SetSpecificBusScanningRangeFuncPtr | 29 |
| 5.1.3.24 GetSpecificBusScanningRangeFuncPtr | 29 |
| 5.1.3.25 getPortStatusFuncPtr | 29 |
| 5.1.3.26 getPortErrorMsgFuncPtr | 29 |
| 5.1.3.27 RegisterReadFuncPtr | 29 |
| 5.1.3.28 RegisterReadU8FuncPtr | 29 |
| 5.1.3.29 RegisterReadS8FuncPtr | 29 |
| 5.1.3.30 RegisterReadU16FuncPtr | 30 |
| 5.1.3.31 RegisterReadS16FuncPtr | 30 |
| 5.1.3.32 RegisterReadU32FuncPtr | 30 |
| 5.1.3.33 RegisterReadS32FuncPtr | 30 |
| 5.1.3.34 RegisterReadU64FuncPtr | 30 |
| 5.1.3.35 RegisterReadS64FuncPtr | 30 |
| 5.1.3.36 RegisterReadF32FuncPtr | 30 |
| 5.1.3.37 RegisterReadF64FuncPtr | 30 |
| 5.1.3.38 RegisterReadAsciiFuncPtr | 30 |
| 5.1.3.39 RegisterWriteFuncPtr | 31 |
| 5.1.3.40 RegisterWriteU8FuncPtr | 31 |
| 5.1.3.41 RegisterWriteS8FuncPtr | 31 |
| 5.1.3.42 RegisterWriteU16FuncPtr | 31 |
| 5.1.3.43 RegisterWriteS16FuncPtr | 31 |
| 5.1.3.44 RegisterWriteU32FuncPtr | 31 |
| 5.1.3.45 RegisterWriteS32FuncPtr | 31 |
| 5.1.3.46 RegisterWriteU64FuncPtr | 31 |
| 5.1.3.47 RegisterWriteS64FuncPtr | 32 |
| 5.1.3.48 RegisterWriteF32FuncPtr | 32 |
| 5.1.3.49 RegisterWriteF64FuncPtr | 32 |
| 5.1.3.50 RegisterWriteAsciiFuncPtr | 32 |
| 5.1.3.51 RegisterWriteReadFuncPtr | 32 |
| 5.1.3.52 RegisterWriteReadU8FuncPtr | 32 |
| 5.1.3.53 RegisterWriteReadS8FuncPtr | 32 |
| 5.1.3.54 RegisterWriteReadU16FuncPtr | 32 |
| 5.1.3.55 RegisterWriteReadS16FuncPtr | 33 |
| 5.1.3.56 RegisterWriteReadU32FuncPtr | 33 |
| 5.1.3.57 RegisterWriteReadS32FuncPtr | 33 |
| 5.1.3.58 RegisterWriteReadU64FuncPtr | 33 |
| 5.1.3.59 RegisterWriteReadS64FuncPtr | 33 |
| 5.1.3.60 RegisterWriteReadF32FuncPtr | 33 |
| 5.1.3.61 RegisterWriteReadF64FuncPtr | 33 |
| 5.1.3.62 RegisterWriteReadAsciiFuncPtr | 33 |
| 5.1.3.63 DeviceGetTypeFuncPtr | 34 |
| 5.1.3.64 DeviceGetTypeV2FuncPtr | 34 |

| | |
|--|----|
| 5.1.3.65 DeviceGetSysTypeFuncPtr | 34 |
| 5.1.3.66 DeviceGetPartNumberStrFuncPtr | 34 |
| 5.1.3.67 DeviceGetPCBVersionFuncPtr | 34 |
| 5.1.3.68 DeviceGetStatusBitsFuncPtr | 34 |
| 5.1.3.69 DeviceGetErrorCodeFuncPtr | 34 |
| 5.1.3.70 DeviceGetBootloaderVersionFuncPtr | 34 |
| 5.1.3.71 DeviceGetBootloaderVersionStrFuncPtr | 34 |
| 5.1.3.72 DeviceGetFirmwareVersionFuncPtr | 35 |
| 5.1.3.73 DeviceGetFirmwareVersionStrFuncPtr | 35 |
| 5.1.3.74 DeviceGetModuleSerialNumberStrFuncPtr | 35 |
| 5.1.3.75 DeviceGetPCBSerialNumberStrFuncPtr | 35 |
| 5.1.3.76 DeviceCreateFuncPtr | 35 |
| 5.1.3.77 DeviceExistsFuncPtr | 35 |
| 5.1.3.78 DeviceRemoveFuncPtr | 35 |
| 5.1.3.79 DeviceRemoveAllFuncPtr | 35 |
| 5.1.3.80 DeviceGetAllTypesFuncPtr | 35 |
| 5.1.3.81 DeviceGetAllTypesV2FuncPtr | 36 |
| 5.1.3.82 DeviceGetModeFuncPtr | 36 |
| 5.1.3.83 DeviceGetLiveFuncPtr | 36 |
| 5.1.3.84 DeviceSetLiveFuncPtr | 36 |
| 5.1.3.85 RegisterCreateFuncPtr | 36 |
| 5.1.3.86 RegisterExistsFuncPtr | 36 |
| 5.1.3.87 RegisterRemoveFuncPtr | 36 |
| 5.1.3.88 RegisterRemoveAllFuncPtr | 36 |
| 5.1.3.89 RegisterGetAllFuncPtr | 36 |
| 5.1.3.90 PortStatusCallbackFuncPtr | 36 |
| 5.1.3.91 SetCallbackPtrPortInfoFuncPtr | 37 |
| 5.1.3.92 DeviceStatusCallbackFuncPtr | 37 |
| 5.1.3.93 SetCallbackPtrDeviceInfoFuncPtr | 37 |
| 5.1.3.94 RegisterStatusCallbackFuncPtr | 38 |
| 5.1.3.95 SetCallbackPtrRegisterInfoFuncPtr | 38 |
| 5.1.3.96 LabViewPortStatusType | 38 |
| 5.1.3.97 SetLVUserEventPortInfoFuncPtr | 38 |
| 5.1.3.98 LabViewDeviceStatusType | 38 |
| 5.1.3.99 SetLVUserEventDeviceInfoFuncPtr | 39 |
| 5.1.3.100 LabViewRegisterStatusType | 39 |
| 5.1.3.101 SetLVUserEventRegisterInfoFuncPtr | 39 |
| 5.1.4 Enumeration Type Documentation | 39 |
| 5.1.4.1 tPortResultTypes | 39 |
| 5.1.4.2 tP2PPortResultTypes | 39 |
| 5.1.4.3 tDeviceResultTypes | 40 |
| 5.1.4.4 tDeviceModeTypes | 40 |

| | |
|--|----|
| 5.1.4.5 tRegisterResultTypes | 40 |
| 5.1.4.6 tRegisterDataTypes | 41 |
| 5.1.4.7 tRegisterPriorityTypes | 42 |
| 5.1.4.8 tPortStatusTypes | 42 |
| 5.1.4.9 tDeviceStatusTypes | 42 |
| 5.1.4.10 tRegisterStatusTypes | 43 |
| 5.1.4.11 tParamSetUnitTypes | 43 |
| 5.1.5 Function Documentation | 45 |
| 5.1.5.1 getAllPorts() | 45 |
| 5.1.5.2 getOpenPorts() | 45 |
| 5.1.5.3 pointToPointPortAdd() | 45 |
| 5.1.5.4 pointToPointPortGet() | 46 |
| 5.1.5.5 pointToPointPortDel() | 47 |
| 5.1.5.6 openPorts() | 47 |
| 5.1.5.7 closePorts() | 48 |
| 5.1.5.8 setLegacyBusScanning() | 48 |
| 5.1.5.9 getLegacyBusScanning() | 49 |
| 5.1.5.10 setSpecificBusScanningRange() | 49 |
| 5.1.5.11 getSpecificBusScanningRange() | 50 |
| 5.1.5.12 getPortStatus() | 50 |
| 5.1.5.13 getPortErrorMsg() | 50 |
| 5.1.5.14 registerRead() | 51 |
| 5.1.5.15 registerReadU8() | 52 |
| 5.1.5.16 registerReadS8() | 52 |
| 5.1.5.17 registerReadU16() | 53 |
| 5.1.5.18 registerReadS16() | 53 |
| 5.1.5.19 registerReadU32() | 54 |
| 5.1.5.20 registerReadS32() | 55 |
| 5.1.5.21 registerReadU64() | 55 |
| 5.1.5.22 registerReadS64() | 56 |
| 5.1.5.23 registerReadF32() | 56 |
| 5.1.5.24 registerReadF64() | 57 |
| 5.1.5.25 registerReadAscii() | 58 |
| 5.1.5.26 registerWrite() | 58 |
| 5.1.5.27 registerWriteU8() | 59 |
| 5.1.5.28 registerWriteS8() | 59 |
| 5.1.5.29 registerWriteU16() | 60 |
| 5.1.5.30 registerWriteS16() | 61 |
| 5.1.5.31 registerWriteU32() | 61 |
| 5.1.5.32 registerWriteS32() | 62 |
| 5.1.5.33 registerWriteU64() | 63 |
| 5.1.5.34 registerWriteS64() | 63 |

| | |
|---|----|
| 5.1.5.35 registerWriteF32() | 64 |
| 5.1.5.36 registerWriteF64() | 64 |
| 5.1.5.37 registerWriteAscii() | 65 |
| 5.1.5.38 registerWriteRead() | 66 |
| 5.1.5.39 registerWriteReadU8() | 67 |
| 5.1.5.40 registerWriteReadS8() | 67 |
| 5.1.5.41 registerWriteReadU16() | 68 |
| 5.1.5.42 registerWriteReadS16() | 68 |
| 5.1.5.43 registerWriteReadU32() | 69 |
| 5.1.5.44 registerWriteReadS32() | 70 |
| 5.1.5.45 registerWriteReadU64() | 70 |
| 5.1.5.46 registerWriteReadS64() | 71 |
| 5.1.5.47 registerWriteReadF32() | 72 |
| 5.1.5.48 registerWriteReadF64() | 72 |
| 5.1.5.49 registerWriteReadAscii() | 73 |
| 5.1.5.50 deviceGetType() | 74 |
| 5.1.5.51 deviceGetTypeV2() | 74 |
| 5.1.5.52 deviceGetSysType() | 75 |
| 5.1.5.53 deviceGetPartNumberStr() | 75 |
| 5.1.5.54 deviceGetPCBVersion() | 76 |
| 5.1.5.55 deviceGetStatusBits() | 76 |
| 5.1.5.56 deviceGetErrorCode() | 77 |
| 5.1.5.57 deviceGetBootloaderVersion() | 77 |
| 5.1.5.58 deviceGetBootloaderVersionStr() | 78 |
| 5.1.5.59 deviceGetFirmwareVersion() | 78 |
| 5.1.5.60 deviceGetFirmwareVersionStr() | 79 |
| 5.1.5.61 deviceGetModuleSerialNumberStr() | 79 |
| 5.1.5.62 deviceGetPCBSerialNumberStr() | 80 |
| 5.1.5.63 deviceCreate() | 81 |
| 5.1.5.64 deviceExists() | 81 |
| 5.1.5.65 deviceRemove() | 82 |
| 5.1.5.66 deviceRemoveAll() | 82 |
| 5.1.5.67 deviceGetAllTypes() | 82 |
| 5.1.5.68 deviceGetAllTypesV2() | 83 |
| 5.1.5.69 deviceGetMode() | 84 |
| 5.1.5.70 deviceGetLive() | 84 |
| 5.1.5.71 deviceSetLive() | 85 |
| 5.1.5.72 registerCreate() | 85 |
| 5.1.5.73 registerExists() | 86 |
| 5.1.5.74 registerRemove() | 86 |
| 5.1.5.75 registerRemoveAll() | 87 |
| 5.1.5.76 registerGetAll() | 87 |

| | |
|---------------------------------------|-----------|
| 5.1.5.77 setCallbackPtrPortInfo() | 88 |
| 5.1.5.78 setCallbackPtrDeviceInfo() | 88 |
| 5.1.5.79 setCallbackPtrRegisterInfo() | 88 |
| 5.1.5.80 setLVUserEventPortInfo() | 88 |
| 5.1.5.81 setLVUserEventDeviceInfo() | 89 |
| 5.1.5.82 setLVUserEventRegisterInfo() | 89 |
| 5.2 NKTPDLL.h | 89 |
| Index | 99 |

Chapter 1

Deprecated List

Global [deviceGetAllTypes](#) (const char *portname, unsigned char *types, unsigned char *maxTypes)

Should not be used in new applications, use [deviceGetAllTypesV2](#) instead.

Global [deviceGetPartNumberStr](#) (const char *portname, const unsigned char devId, char *partnumber, unsigned char *maxLen)

No longer used - Should not be used in new applications.

Global [deviceGetType](#) (const char *portname, const unsigned char devId, unsigned char *devType)

Should not be used in new applications, use [deviceGetTypeV2](#) instead.

Global [DevicePartNumberChanged](#)

No longer available.

Global [DevModeError](#)

No longer used.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

| | |
|--|----|
| LvDeviceStatusStruct | |
| LvDeviceStatusStruct, A LabView userevent data package | 7 |
| LvPortStatusStruct | |
| LvPortStatusStruct, A LabView userevent data package | 8 |
| LvRegisterStatusStruct | |
| LvRegisterStatusStruct, A LabView userevent data package | 9 |
| tDateTimeStruct | |
| The tDateTime struct 24 hour format | 11 |
| tParamSetStruct | |
| The tParameterSet struct | 12 |

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

[NKTPDLL.h](#)

NKTP DLL Interface, a communication DLL for interfacing to NKT Photonics products being controlled via the Interbus protocol. The NKTPDLL abstracts the burden of telegram creation and communication handling when communicating/controlling the NKT Photonics products . . . [15](#)

Chapter 4

Data Structure Documentation

4.1 lvDeviceStatusStruct Struct Reference

[lvDeviceStatusStruct](#), A LabView userevent data package

```
#include <NKTPDLL.h>
```

Data Fields

- char [portname](#) [32]
Zero terminated string giving the originating portname.
- unsigned char [devId](#)
The originating device id (module address).
- [DeviceStatusTypes](#) [status](#)
The current port status as [tDeviceStatusTypes](#).
- unsigned char [devDataLen](#)
Number of databytes in devData.
- unsigned char [devData](#) [255]
device data as specified in status.

4.1.1 Detailed Description

[lvDeviceStatusStruct](#), A LabView userevent data package

4.1.2 Field Documentation

4.1.2.1 portname

```
char lvDeviceStatusStruct::portname[32]
```

Zero terminated string giving the originating portname.

4.1.2.2 devId

```
unsigned char lvDeviceStatusStruct::devId
```

The originating device id (module address).

4.1.2.3 status

```
DeviceStatusTypes lvDeviceStatusStruct::status
```

The current port status as [tDeviceStatusTypes](#).

4.1.2.4 devDataLen

```
unsigned char lvDeviceStatusStruct::devDataLen
```

Number of databytes in devData.

4.1.2.5 devData

```
unsigned char lvDeviceStatusStruct::devData[255]
```

device data as specified in status.

4.2 lvPortStatusStruct Struct Reference

[lvPortStatusStruct](#), A LabView usevent data package

```
#include <NKTPDLL.h>
```

Data Fields

- char [portname](#) [32]
Zero terminated string giving the originating portname.
- [PortStatusTypes](#) [status](#)
The current port status as [tPortStatusTypes](#).
- unsigned char [curScanAdr](#)
When status is [PortScanProgress](#) or [PortScanDeviceFound](#) this indicates the current module address scanned or found.
- unsigned char [maxScanAdr](#)
When status is [PortScanProgress](#) or [PortScanDeviceFound](#) this indicates the last module address to be scanned.
- unsigned short [foundType](#)
When status is [PortScanDeviceFound](#) this value will represent the found module type.

4.2.1 Detailed Description

[lvPortStatusStruct](#), A LabView usevent data package

4.2.2 Field Documentation

4.2.2.1 portname

```
char lvPortStatusStruct::portname[32]
```

Zero terminated string giving the originating portname.

4.2.2.2 status

```
PortStatusTypes lvPortStatusStruct::status
```

The current port status as [tPortStatusTypes](#).

4.2.2.3 curScanAdr

```
unsigned char lvPortStatusStruct::curScanAdr
```

When status is [PortScanProgress](#) or [PortScanDeviceFound](#) this indicates the current module address scanned or found.

4.2.2.4 maxScanAdr

```
unsigned char lvPortStatusStruct::maxScanAdr
```

When status is [PortScanProgress](#) or [PortScanDeviceFound](#) this indicates the last module address to be scanned.

4.2.2.5 foundType

```
unsigned short lvPortStatusStruct::foundType
```

When status is [PortScanDeviceFound](#) this value will represent the found module type.

Note

Was originally a byte, but since we now has moduletypes above 0xFF, this has been changed to a short.

4.3 lvRegisterStatusStruct Struct Reference

[lvRegisterStatusStruct](#), A LabView userevent data package

```
#include <NKTPDLL.h>
```

Data Fields

- char [portname](#) [32]
Zero terminated string giving the originating portname.
- unsigned char [devId](#)
The originating device id (module address).
- unsigned char [regId](#)
The originating register id.
- [RegisterStatusTypes](#) [status](#)
The current register status as a [tRegisterStatusTypes](#) value.
- [RegisterDataTypes](#) [regType](#)
The [tRegisterDataTypes](#), not used internally but could be used in a common callback function to determine data type. Set when the register is created with [registerCreate](#).
- unsigned char [regDataLen](#)
Number of databytes.
- unsigned char [regData](#) [255]
The register data.

4.3.1 Detailed Description

[lvRegisterStatusStruct](#), A LabView userevent data package

4.3.2 Field Documentation

4.3.2.1 portname

```
char lvRegisterStatusStruct::portname[32]
```

Zero terminated string giving the originating portname.

4.3.2.2 devId

```
unsigned char lvRegisterStatusStruct::devId
```

The originating device id (module address).

4.3.2.3 regId

```
unsigned char lvRegisterStatusStruct::regId
```

The originating register id.

4.3.2.4 status

```
RegisterStatusTypes lvRegisterStatusStruct::status
```

The current register status as a [tRegisterStatusTypes](#) value.

4.3.2.5 regType

`RegisterDataTypes lvRegisterStatusStruct::regType`

The `tRegisterDataTypes`, not used internally but could be used in a common callback function to determine data type. Set when the register is created with `registerCreate`.

4.3.2.6 regDataLen

`unsigned char lvRegisterStatusStruct::regDataLen`

Number of databytes.

4.3.2.7 regData

`unsigned char lvRegisterStatusStruct::regData[255]`

The register data.

4.4 tDateTimeStruct Struct Reference

The tDateTime struct 24 hour format.

```
#include <NKTPDLL.h>
```

Data Fields

- unsigned char `Sec`
Seconds.
- unsigned char `Min`
Minutes.
- unsigned char `Hour`
Hours.
- unsigned char `Day`
Day.
- unsigned char `Month`
Months.
- unsigned char `Year`
Years.

4.4.1 Detailed Description

The tDateTime struct 24 hour format.

4.4.2 Field Documentation

4.4.2.1 Sec

```
unsigned char tDateTimeStruct::Sec
```

Seconds.

4.4.2.2 Min

```
unsigned char tDateTimeStruct::Min
```

Minutes.

4.4.2.3 Hour

```
unsigned char tDateTimeStruct::Hour
```

Hours.

4.4.2.4 Day

```
unsigned char tDateTimeStruct::Day
```

Day.

4.4.2.5 Month

```
unsigned char tDateTimeStruct::Month
```

Months.

4.4.2.6 Year

```
unsigned char tDateTimeStruct::Year
```

Years.

4.5 tParamSetStruct Struct Reference

The tParameterSet struct.

```
#include <NKTPDLL.h>
```

Data Fields

- [ParamSetUnitTypes](#) Unit
Unit type as defined in [tParamSetUnitTypes](#).
- unsigned char [ErrorHandler](#)
Warning/Errorhandler not used.
- unsigned short [StartVal](#)
Setpoint for Settings parameterset, unused in Measurement parametersets.
- unsigned short [FactoryVal](#)
Factory Setpoint for Settings parameterset, unused in Measurement parametersets.
- unsigned short [ULimit](#)
Upper limit.
- unsigned short [LLimit](#)
Lower limit.
- signed short [Numerator](#)
[Numerator\(X\)](#) for calculation.
- signed short [Denominator](#)
[Denominator\(Y\)](#) for calculation.
- signed short [Offset](#)
Offset for calculation.

4.5.1 Detailed Description

The tParameterSet struct.

Note

This is how calculation on parametersets is done internally by modules:

$DAC_value = (value * (X/Y)) + Offset$; Where value is either [ParameterSetType::StartVal](#) or [ParameterSetType::FactoryVal](#)
 $value = (ADC_value * (X/Y)) + Offset$; Where value often is available via another measurement register

4.5.2 Field Documentation

4.5.2.1 Unit

```
ParamSetUnitTypes tParamSetStruct::Unit
```

Unit type as defined in [tParamSetUnitTypes](#).

4.5.2.2 ErrorHandler

```
unsigned char tParamSetStruct::ErrorHandler
```

Warning/Errorhandler not used.

4.5.2.3 StartVal

```
unsigned short tParamSetStruct::StartVal
```

Setpoint for Settings parameterset, unused in Measurement parametersets.

4.5.2.4 FactoryVal

```
unsigned short tParamSetStruct::FactoryVal
```

Factory Setpoint for Settings parameterset, unused in Measurement parametersets.

4.5.2.5 ULimit

```
unsigned short tParamSetStruct::ULimit
```

Upper limit.

4.5.2.6 LLimit

```
unsigned short tParamSetStruct::LLimit
```

Lower limit.

4.5.2.7 Numerator

```
signed short tParamSetStruct::Numerator
```

[Numerator\(X\)](#) for calculation.

4.5.2.8 Denominator

```
signed short tParamSetStruct::Denominator
```

[Denominator\(Y\)](#) for calculation.

4.5.2.9 Offset

```
signed short tParamSetStruct::Offset
```

Offset for calculation.

Chapter 5

File Documentation

5.1 NKTPDLL.h File Reference

NKTP DLL Interface, a communication DLL for interfacing to NKT Photonics products being controlled via the Interbus protocol. The NKTPDLL abstracts the burden of telegram creation and communication handling when communicating/controlling the NKT Photonics products.

Data Structures

- struct [tDateTimeStruct](#)
The tDateTime struct 24 hour format.
- struct [tParamSetStruct](#)
The tParameterSet struct.
- struct [lvPortStatusStruct](#)
lvPortStatusStruct, A LabView userevent data package
- struct [lvDeviceStatusStruct](#)
lvDeviceStatusStruct, A LabView userevent data package
- struct [lvRegisterStatusStruct](#)
lvRegisterStatusStruct, A LabView userevent data package

Macros

- #define [NKTPDLL_EXPORT](#) __declspec(dllexport)

Typedefs

- typedef unsigned char [PortResultTypes](#)
- typedef unsigned char [P2PPortResultTypes](#)
- typedef unsigned char [DeviceResultTypes](#)
- typedef unsigned char [DeviceModeTypes](#)
- typedef unsigned char [RegisterResultTypes](#)
- typedef unsigned char [RegisterDataTypes](#)
- typedef unsigned char [RegisterPriorityTypes](#)
- typedef unsigned char [PortStatusTypes](#)
- typedef unsigned char [DeviceStatusTypes](#)
- typedef unsigned char [RegisterStatusTypes](#)
- typedef struct [tDateTimeStruct](#) [DateTimeType](#)
The tDateTime struct 24 hour format.
- typedef unsigned char [ParamSetUnitTypes](#)
- typedef struct [tParamSetStruct](#) [ParameterSetType](#)
The tParameterSet struct.

Enumerations

- enum `tPortResultTypes` {
`OPSuccess = 0` , `OPFailed = 1` , `OPPortNotFound = 2` , `OPNoDevices = 3` ,
`OPApplicationBusy = 4` }
The tPortResultTypes enum.
- enum `tP2PPortResultTypes` {
`P2PSuccess = 0` , `P2PInvalidPortname = 1` , `P2PInvalidLocalIP = 2` , `P2PInvalidRemoteIP = 3` ,
`P2PPortnameNotFound = 4` , `P2PPortnameExists = 5` , `P2PApplicationBusy = 6` }
The tPointToPointPortStatus enum.
- enum `tDeviceResultTypes` {
`DevResultSuccess = 0` , `DevResultWaitTimeout = 1` , `DevResultFailed = 2` , `DevResultDeviceNotFound = 3` ,
`DevResultPortNotFound = 4` , `DevResultPortOpenError = 5` , `DevResultApplicationBusy = 6` }
The tDeviceResultTypes enum.
- enum `tDeviceModeTypes` {
`DevModeDisabled = 0` , `DevModeAnalyzeInit = 1` , `DevModeAnalyze = 2` , `DevModeNormal = 3` ,
`DevModeLogDownload = 4` , `DevModeError = 5` , `DevModeTimeout = 6` , `DevModeUpload = 7` }
The tDeviceModeTypes enum.
- enum `tRegisterResultTypes` {
`RegResultSuccess = 0` , `RegResultReadError = 1` , `RegResultFailed = 2` , `RegResultBusy = 3` ,
`RegResultNacked = 4` , `RegResultCRCError = 5` , `RegResultTimeout = 6` , `RegResultComError = 7` ,
`RegResultTypeError = 8` , `RegResultIndexError = 9` , `RegResultPortClosed = 10` , `RegResultRegisterNotFound`
`= 11` ,
`RegResultDeviceNotFound = 12` , `RegResultPortNotFound = 13` , `RegResultPortOpenError = 14` ,
`RegResultApplicationBusy = 15` }
The tRegisterResultTypes enum.
- enum `tRegisterDataTypes` {
`RegData_Unknown = 0` , `RegData_Mixed = 1` , `RegData_U8 = 2` , `RegData_S8 = 3` ,
`RegData_U16 = 4` , `RegData_S16 = 5` , `RegData_U32 = 6` , `RegData_S32 = 7` ,
`RegData_F32 = 8` , `RegData_U64 = 9` , `RegData_S64 = 10` , `RegData_F64 = 11` ,
`RegData_Ascii = 12` , `RegData_Paramset = 13` , `RegData_B8 = 14` , `RegData_H8 = 15` ,
`RegData_B16 = 16` , `RegData_H16 = 17` , `RegData_B32 = 18` , `RegData_H32 = 19` ,
`RegData_B64 = 20` , `RegData_H64 = 21` , `RegData_DateTime = 22` }
The tRegisterDataTypes enum.
- enum `tRegisterPriorityTypes` { `RegPriority_Low = 0` , `RegPriority_High = 1` }
The tRegisterPriorityTypes enum.
- enum `tPortStatusTypes` {
`PortStatusUnknown = 0` , `PortOpening = 1` , `PortOpened = 2` , `PortOpenFail = 3` ,
`PortScanStarted = 4` , `PortScanProgress = 5` , `PortScanDeviceFound = 6` , `PortScanEnded = 7` ,
`PortClosing = 8` , `PortClosed = 9` , `PortReady = 10` }
The tPortStatusTypes enum.
- enum `tDeviceStatusTypes` {
`DeviceModeChanged = 0` , `DeviceLiveChanged = 1` , `DeviceTypeChanged = 2` , `DevicePartNumberChanged`
`= 3` ,
`DevicePCBVersionChanged = 4` , `DeviceStatusBitsChanged = 5` , `DeviceErrorCodeChanged = 6` ,
`DeviceBIVerChanged = 7` ,
`DeviceFwVerChanged = 8` , `DeviceModuleSerialChanged = 9` , `DevicePCBSerialChanged = 10` ,
`DeviceSysTypeChanged = 11` }
The tDeviceStatusTypes enum.
- enum `tRegisterStatusTypes` {
`RegSuccess = 0` , `RegBusy = 1` , `RegNacked = 2` , `RegCRCError = 3` ,
`RegTimeout = 4` , `RegComError = 5` }
The tRegisterStatusTypes enum.

- enum `tParamSetUnitTypes` {
`UnitNone` = 0 , `UnitmV` = 1 , `UnitV` = 2 , `UnituA` = 3 ,
`UnitmA` = 4 , `UnitA` = 5 , `UnituW` = 6 , `UnitcmW` = 7 ,
`UnitdmW` = 8 , `UnitmW` = 9 , `UnitW` = 10 , `UnitmC` = 11 ,
`UnitcC` = 12 , `UnitdC` = 13 , `Unitpm` = 14 , `Unitdm` = 15 ,
`Unitnm` = 16 , `UnitPerCent` = 17 , `UnitPerMille` = 18 , `UnitcmA` = 19 ,
`UnitdmA` = 20 , `UnitRPM` = 21 , `UnitdBm` = 22 , `UnitcBm` = 23 ,
`UnitmBm` = 24 , `UnitdB` = 25 , `UnitcB` = 26 , `UnitmB` = 27 ,
`Unitdpm` = 28 , `UnitcV` = 29 , `UnitdV` = 30 , `Unitlm` = 31 ,
`Unitdlm` = 32 , `Unitclm` = 33 , `Unitmlm` = 34 , `UnitHz` = 35 ,
`UnitkHz` = 36 , `UnitMHz` = 37 , `UnitSec` = 38 , `UnitmSec` = 39 ,
`UnituSec` = 40 , `UnitdA` = 41 , `UnitcA` = 42 , `UnitduA` = 43 ,
`UnitcuA` = 44 , `UnitnA` = 45 , `UnitdW` = 46 , `UnitcW` = 47 ,
`UnitpA` = 48 }

The tParamSetUnitTypes enum.

Port functions

- typedef void(__cdecl * `GetAllPortsFuncPtr`) (char *portnames, unsigned short *maxLen)
- typedef void(__cdecl * `GetOpenPortsFuncPtr`) (char *portnames, unsigned short *maxLen)
- typedef `P2PPortResultTypes`(__cdecl * `PointToPointPortAddFuncPtr`) (const char *portname, const char *hostAddress, const unsigned short hostPort, const char *clientAddress, const unsigned short clientPort, const unsigned char protocol, const unsigned char msTimeout)
- typedef `P2PPortResultTypes`(__cdecl * `PointToPointPortGetFuncPtr`) (const char *portname, char *hostAddress, unsigned char *hostMaxLen, unsigned short *hostPort, char *clientAddress, unsigned char *clientMaxLen, unsigned short *clientPort, unsigned char *protocol, unsigned char *msTimeout)
- typedef `P2PPortResultTypes`(__cdecl * `PointToPointPortDelFuncPtr`) (const char *portname)
- typedef `PortResultTypes`(__cdecl * `OpenPortsFuncPtr`) (const char *portnames, const char autoMode, const char liveMode)
- typedef `PortResultTypes`(__cdecl * `ClosePortsFuncPtr`) (const char *portnames)
- typedef void(__cdecl * `SetLegacyBusScanningFuncPtr`) (const char legacyScanning)
- typedef unsigned char(__cdecl * `GetLegacyBusScanningFuncPtr`) ()
- typedef void(__cdecl * `SetSpecificBusScanningRangeFuncPtr`) (const char specificScanning, const unsigned char startAddress, const unsigned char endAddress)
- typedef unsigned char(__cdecl * `GetSpecificBusScanningRangeFuncPtr`) (char *specificScanning, unsigned char *startAddress, unsigned char *endAddress)
- typedef `PortResultTypes`(__cdecl * `getPortStatusFuncPtr`) (const char *portname, `PortStatusTypes` *portStatus)
- typedef `PortResultTypes`(__cdecl * `getPortErrorMsgFuncPtr`) (const char *portname, char *errorMessage, unsigned short *maxLen)
- `NKTPDLL_EXPORT` void `getAllPorts` (char *portnames, unsigned short *maxLen)
Returns a comma separated string with all existing ports.
- `NKTPDLL_EXPORT` void `getOpenPorts` (char *portnames, unsigned short *maxLen)
Returns a comma separated string with all already opened ports.
- `NKTPDLL_EXPORT` `P2PPortResultTypes` `pointToPointPortAdd` (const char *portname, const char *hostAddress, const unsigned short hostPort, const char *clientAddress, const unsigned short clientPort, const unsigned char protocol, const unsigned char msTimeout)
Creates or Modifies a point to point port.
- `NKTPDLL_EXPORT` `P2PPortResultTypes` `pointToPointPortGet` (const char *portname, char *hostAddress, unsigned char *hostMaxLen, unsigned short *hostPort, char *clientAddress, unsigned char *clientMaxLen, unsigned short *clientPort, unsigned char *protocol, unsigned char *msTimeout)
Retrieve an already created point to point port setting.
- `NKTPDLL_EXPORT` `P2PPortResultTypes` `pointToPointPortDel` (const char *portname)
Delete an already created point to point port.

- **NKTPDLL_EXPORT PortResultTypes openPorts** (const char *portnames, const char autoMode, const char liveMode)
Opens the provided portname(s), or all available ports if an empty string provided. Repeatedly calls is allowed to reopen and/or rescan for devices.
- **NKTPDLL_EXPORT PortResultTypes closePorts** (const char *portnames)
Closes the provided portname(s), or all opened ports if an empty string provided.
- **NKTPDLL_EXPORT void setLegacyBusScanning** (const char legacyScanning)
Sets legacy busscanning on or off.
- **NKTPDLL_EXPORT unsigned char getLegacyBusScanning** ()
Gets legacy busscanning status.
- **NKTPDLL_EXPORT void setSpecificBusScanningRange** (const char specificScanning, const unsigned char startAddress, const unsigned char endAddress)
Sets specific busscanning address range, on/off.
- **NKTPDLL_EXPORT void getSpecificBusScanningRange** (char *specificScanning, unsigned char *startAddress, unsigned char *endAddress)
Gets specific busscanning address range and status.
- **NKTPDLL_EXPORT PortResultTypes getPortStatus** (const char *portname, **PortStatusTypes** *portStatus)
Retrieve tPortStatusTypes for a given port.
- **NKTPDLL_EXPORT PortResultTypes getPortErrorMsg** (const char *portname, char *errorMessage, unsigned short *maxLen)
Retrieve error message for a given port. An empty string indicates no error.

Dedicated - Register read functions.

It is not necessary to open the port, create the device or register before using those functions, since they will do a dedicated action. Even though an already opened port would be preferred in time critical situations where a lot of reads or writes is required.

- typedef **RegisterResultTypes**(__cdecl * **RegisterReadFuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, void *readData, unsigned char *readSize, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadU8FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned char *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadS8FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, signed char *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadU16FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned short *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadS16FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, signed short *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadU32FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned long *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadS32FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, signed long *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadU64FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned long long *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadS64FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, signed long long *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadF32FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, float *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadF64FuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, double *value, const short index)
- typedef **RegisterResultTypes**(__cdecl * **RegisterReadAsciiFuncPtr**) (const char *portname, const unsigned char devId, const unsigned char regId, char *readStr, unsigned char *maxLen, const short index)

- [NKTPDLL_EXPORT RegisterResultTypes registerRead](#) (const char *portname, const unsigned char devId, const unsigned char regId, void *readData, unsigned char *readSize, const short index)
Reads a register value and returns the result in readData area.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadU8](#) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned char *value, const short index)
Reads an unsigned char (8bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadS8](#) (const char *portname, const unsigned char devId, const unsigned char regId, signed char *value, const short index)
Reads a signed char (8bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadU16](#) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned short *value, const short index)
Reads an unsigned short (16bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadS16](#) (const char *portname, const unsigned char devId, const unsigned char regId, signed short *value, const short index)
Reads a signed short (16bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadU32](#) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned long *value, const short index)
Reads an unsigned long (32bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadS32](#) (const char *portname, const unsigned char devId, const unsigned char regId, signed long *value, const short index)
Reads a signed long (32bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadU64](#) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned long long *value, const short index)
Reads an unsigned long long (64bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadS64](#) (const char *portname, const unsigned char devId, const unsigned char regId, signed long long *value, const short index)
Reads a signed long long (64bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadF32](#) (const char *portname, const unsigned char devId, const unsigned char regId, float *value, const short index)
Reads a float (32bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadF64](#) (const char *portname, const unsigned char devId, const unsigned char regId, double *value, const short index)
Reads a double (64bit) register value and returns the result in value.
- [NKTPDLL_EXPORT RegisterResultTypes registerReadAscii](#) (const char *portname, const unsigned char devId, const unsigned char regId, char *readStr, unsigned char *maxLen, const short index)
Reads a Ascii string register value and returns the result in readStr area.

Dedicated - Register write functions.

It is not necessary to open the port, create the device or register before using those functions, since they will do a dedicated action. Even though an already opened port would be preferred in time critical situations where a lot of reads or writes is required.

- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const void *writeData, const unsigned char writeSize, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteU8FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned char value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteS8FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed char value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteU16FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned short value, const short index)

- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteS16FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed short value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteU32FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteS32FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteU64FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long long value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteS64FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long long value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteF32FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const float value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteF64FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const double value, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteAsciiFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const char *writeStr, const char writeEOL, const short index)
- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWrite](#) (const char *portname, const unsigned char devId, const unsigned char regId, const void *writeData, const unsigned char writeSize, const short index)

Writes a register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteU8](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned char value, const short index)

Writes an unsigned char (8bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteS8](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed char value, const short index)

Writes a signed char (8bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteU16](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned short value, const short index)

Writes an unsigned short (16bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteS16](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed short value, const short index)

Writes a signed short (16bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteU32](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long value, const short index)

Writes an unsigned long (32bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteS32](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long value, const short index)

Writes a signed long (32bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteU64](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long long value, const short index)

Writes an unsigned long long (64bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteS64](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long long value, const short index)

Writes a signed long long (64bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteF32](#) (const char *portname, const unsigned char devId, const unsigned char regId, const float value, const short index)

Writes a float (32bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteF64](#) (const char *portname, const unsigned char devId, const unsigned char regId, const double value, const short index)

Writes a double (64bit) register value.

- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerWriteAscii](#) (const char *portname, const unsigned char devId, const unsigned char regId, const char *writeStr, const char writeEOL, const short index)

Writes a string register value.

Dedicated - Register write/read functions (A write immediately followed by a read)

It is not necessary to open the port, create the device or register before using those functions, since they will do a dedicated action. Even though an already opened port would be preferred in time critical situations where a lot of reads or writes is required.

- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const void *writeData, const unsigned char writeSize, void *readData, unsigned char *readSize, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadU8FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned char writeValue, unsigned char *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadS8FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed char writeValue, signed char *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadU16FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned short writeValue, unsigned short *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadS16FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed short writeValue, signed short *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadU32FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long writeValue, unsigned long *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadS32FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long writeValue, signed long *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadU64FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long long writeValue, unsigned long long *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadS64FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long long writeValue, signed long long *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadF32FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const float writeValue, float *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadF64FuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const double writeValue, double *readValue, const short index)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterWriteReadAsciiFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const char *writeStr, const char writeEOL, char *readStr, unsigned char *maxLen, const short index)
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteRead](#) (const char *portname, const unsigned char devId, const unsigned char regId, const void *writeData, const unsigned char writeSize, void *readData, unsigned char *readSize, const short index)

Writes and Reads a register value before returning.

- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU8](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned char writeValue, unsigned char *readValue, const short index)

Writes and Reads an unsigned char (8bit) register value.

- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS8](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed char writeValue, signed char *readValue, const short index)

Writes and Reads a signed char (8bit) register value.

- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU16](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned short writeValue, unsigned short *readValue, const short index)

Writes and Reads an unsigned short (16bit) register value.

- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS16](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed short writeValue, signed short *readValue, const short index)
Writes and Reads a signed short (16bit) register value.
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU32](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long writeValue, unsigned long *readValue, const short index)
Writes and Reads an unsigned long (32bit) register value.
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS32](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long writeValue, signed long *readValue, const short index)
Writes and Reads a signed long (32bit) register value.
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU64](#) (const char *portname, const unsigned char devId, const unsigned char regId, const unsigned long long writeValue, unsigned long long *readValue, const short index)
Writes and Reads an unsigned long long (64bit) register value.
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS64](#) (const char *portname, const unsigned char devId, const unsigned char regId, const signed long long writeValue, signed long long *readValue, const short index)
Writes and Reads a signed long long (64bit) register value.
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadF32](#) (const char *portname, const unsigned char devId, const unsigned char regId, const float writeValue, float *readValue, const short index)
Writes and Reads a float (32bit) register value.
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadF64](#) (const char *portname, const unsigned char devId, const unsigned char regId, const double writeValue, double *readValue, const short index)
Writes and Reads a double (64bit) register value.
- [NKTPDLL_EXPORT RegisterResultTypes registerWriteReadAscii](#) (const char *portname, const unsigned char devId, const unsigned char regId, const char *writeStr, const char writeEOL, char *readStr, unsigned char *maxLen, const short index)
Writes and Reads a string register value.

Dedicated - Device functions

Dedicated - Device functions could be used directly.

It is not necessary to open the port, create the device or register before using those functions, since they will do a dedicated action. Even though an already opened port would be preferred in time critical situations where a lot of reads is required.

- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetTypeFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned char *devType)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetTypeV2FuncPtr](#)) (const char *portname, const unsigned char devId, unsigned short *devType)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetSysTypeFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned char *sysType)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetPartNumberStrFuncPtr](#)) (const char *portname, const unsigned char devId, char *partnumber, unsigned char *maxLen)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetPCBVersionFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned char *PCBVersion)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetStatusBitsFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned long *statusBits)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetErrorCodeFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned short *errorCode)

- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetBootloaderVersionFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned short *version)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetBootloaderVersionStrFuncPtr](#)) (const char *portname, const unsigned char devId, char *versionStr, unsigned char *maxLen)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetFirmwareVersionFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned short *version)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetFirmwareVersionStrFuncPtr](#)) (const char *portname, const unsigned char devId, char *versionStr, unsigned char *maxLen)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetModuleSerialNumberStrFuncPtr](#)) (const char *portname, const unsigned char devId, char *serialNumber, unsigned char *maxLen)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetPCBSerialNumberStrFuncPtr](#)) (const char *portname, const unsigned char devId, char *serialNumber, unsigned char *maxLen)
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetType](#) (const char *portname, const unsigned char devId, unsigned char *devType)
Returns the module type for a specific device id (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetTypeV2](#) (const char *portname, const unsigned char devId, unsigned short *devType)
Returns the module type for a specific device id (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetSysType](#) (const char *portname, const unsigned char devId, unsigned char *sysType)
Returns the system type for a specific device id (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetPartNumberStr](#) (const char *portname, const unsigned char devId, char *partnumber, unsigned char *maxLen)
Returns the partnumber for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetPCBVersion](#) (const char *portname, const unsigned char devId, unsigned char *PCBVersion)
Returns the PCB version for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetStatusBits](#) (const char *portname, const unsigned char devId, unsigned long *statusBits)
Returns the status bits for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetErrorCode](#) (const char *portname, const unsigned char devId, unsigned short *errorCode)
Returns the error code for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetBootloaderVersion](#) (const char *portname, const unsigned char devId, unsigned short *version)
Returns the bootloader version for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetBootloaderVersionStr](#) (const char *portname, const unsigned char devId, char *versionStr, unsigned char *maxLen)
Returns the bootloader version (string) for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetFirmwareVersion](#) (const char *portname, const unsigned char devId, unsigned short *version)
Returns the firmware version for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetFirmwareVersionStr](#) (const char *portname, const unsigned char devId, char *versionStr, unsigned char *maxLen)
Returns the firmware version (string) for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetModuleSerialNumberStr](#) (const char *portname, const unsigned char devId, char *serialNumber, unsigned char *maxLen)
Returns the Module serialnumber (string) for a given device (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetPCBSerialNumberStr](#) (const char *portname, const unsigned char devId, char *serialNumber, unsigned char *maxLen)
Returns the PCB serialnumber (string) for a given device (module address).

Callback - Device functions

Device functions primarily used in callback environments.

- typedef [DeviceResultTypes](#)(__cdecl * [DeviceCreateFuncPtr](#)) (const char *portname, const unsigned char devId, const char waitReady)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceExistsFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned char *exists)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceRemoveFuncPtr](#)) (const char *portname, const unsigned char devId)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceRemoveAllFuncPtr](#)) (const char *portname)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetAllTypesFuncPtr](#)) (const char *portname, unsigned char *types, unsigned char *maxTypes)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetAllTypesV2FuncPtr](#)) (const char *portname, unsigned short *types, unsigned char *maxTypes)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetModeFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned char *devMode)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceGetLiveFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned char *liveMode)
- typedef [DeviceResultTypes](#)(__cdecl * [DeviceSetLiveFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char liveMode)
- [NKTPDLL_EXPORT DeviceResultTypes deviceCreate](#) (const char *portname, const unsigned char devId, const char waitReady)
Creates a device in the internal devicelist. If the [openPorts](#) function has been called with the liveMode = 1 the kernel immediately starts to monitor the device.
- [NKTPDLL_EXPORT DeviceResultTypes deviceExists](#) (const char *portname, const unsigned char devId, unsigned char *exists)
Checks if a specific device already exists in the internal devicelist.
- [NKTPDLL_EXPORT DeviceResultTypes deviceRemove](#) (const char *portname, const unsigned char devId)
Remove a specific device from the internal devicelist.
- [NKTPDLL_EXPORT DeviceResultTypes deviceRemoveAll](#) (const char *portname)
Remove all devices from the internal devicelist. No confirmation given, the list is simply cleared.
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetAllTypes](#) (const char *portname, unsigned char *types, unsigned char *maxTypes)
Returns a list with device types (module types) from the internal devicelist.
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetAllTypesV2](#) (const char *portname, unsigned short *types, unsigned char *maxTypes)
Returns a list with device types (module types) from the internal devicelist.
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetMode](#) (const char *portname, const unsigned char devId, unsigned char *devMode)
Returns the internal device mode for a specific device id (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceGetLive](#) (const char *portname, const unsigned char devId, unsigned char *liveMode)
Returns the internal device live status for a specific device id (module address).
- [NKTPDLL_EXPORT DeviceResultTypes deviceSetLive](#) (const char *portname, const unsigned char devId, const unsigned char liveMode)
Sets the internal device live status for a specific device id (module address).

Callback - Register functions

Device functions primarily used in callback environments.

- typedef [RegisterResultTypes](#)(__cdecl * [RegisterCreateFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const [RegisterPriorityTypes](#) priority, const [RegisterDataTypes](#) dataType)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterExistsFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned char *exists)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterRemoveFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterRemoveAllFuncPtr](#)) (const char *portname, const unsigned char devId)
- typedef [RegisterResultTypes](#)(__cdecl * [RegisterGetAllFuncPtr](#)) (const char *portname, const unsigned char devId, unsigned char *regs, unsigned char *maxRegs)
- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerCreate](#) (const char *portname, const unsigned char devId, const unsigned char regId, const [RegisterPriorityTypes](#) priority, const [RegisterDataTypes](#) dataType)
Creates a register in the internal registerlist. If the [openPorts](#) function has been called with the liveMode = 1 the kernel immediately starts to monitor the register.
- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerExists](#) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned char *exists)
Checks if a specific register already exists in the internal registerlist.
- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerRemove](#) (const char *portname, const unsigned char devId, const unsigned char regId)
Remove a specific register from the internal registerlist.
- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerRemoveAll](#) (const char *portname, const unsigned char devId)
Remove all registers from the internal registerlist. No confirmation given, the list is simply cleared.
- [NKTPDLL_EXPORT](#) [RegisterResultTypes](#) [registerGetAll](#) (const char *portname, const unsigned char devId, unsigned char *regs, unsigned char *maxRegs)
Returns a list with register ids (register addresses) from the internal registerlist.

Callback - Support functions

- typedef void(__cdecl * [PortStatusCallbackFuncPtr](#)) (const char *portname, const [PortStatusTypes](#) status, const unsigned char curScanAdr, const unsigned char maxScanAdr, const unsigned short foundType)
Defines the PortStatusCallbackFuncPtr for the [openPorts](#) and [closePorts](#) functions.
- typedef void(__cdecl * [SetCallbackPtrPortInfoFuncPtr](#)) ([PortStatusCallbackFuncPtr](#) callback)
- typedef void(__cdecl * [DeviceStatusCallbackFuncPtr](#)) (const char *portname, const unsigned char devId, const [DeviceStatusTypes](#) status, const unsigned char devDataLen, const void *devData)
Defines the DeviceStatusCallbackFuncPtr for the devices created with the [deviceCreate](#) function or created automatically via the [openPorts](#) function (Having autoMode = 1).
- typedef void(__cdecl * [SetCallbackPtrDeviceInfoFuncPtr](#)) ([DeviceStatusCallbackFuncPtr](#) callback)
- typedef void(__cdecl * [RegisterStatusCallbackFuncPtr](#)) (const char *portname, const unsigned char devId, const unsigned char regId, const [RegisterStatusTypes](#) status, const [RegisterDataTypes](#) regType, const unsigned char regDataLen, const void *regData)
Defines the RegisterStatusCallbackFuncPtr for the registers created or connected with the [registerCreate](#) function.
- typedef void(__cdecl * [SetCallbackPtrRegisterInfoFuncPtr](#)) ([RegisterStatusCallbackFuncPtr](#) callback)
- [NKTPDLL_EXPORT](#) void [setCallbackPtrPortInfo](#) ([PortStatusCallbackFuncPtr](#) callback)
Enables/Disables callback for port status changes.
- [NKTPDLL_EXPORT](#) void [setCallbackPtrDeviceInfo](#) ([DeviceStatusCallbackFuncPtr](#) callback)
Enables/Disables callback for device status changes.
- [NKTPDLL_EXPORT](#) void [setCallbackPtrRegisterInfo](#) ([RegisterStatusCallbackFuncPtr](#) callback)
Enables/Disables callback for register status changes.

LabView - Support functions

- typedef struct [lvPortStatusStruct](#) [LabViewPortStatusType](#)
lvPortStatusStruct, A LabView userevent data package
- typedef void(__cdecl * [SetLVUserEventPortInfoFuncPtr](#)) (unsigned long *lvUserEventRef)
- typedef struct [lvDeviceStatusStruct](#) [LabViewDeviceStatusType](#)
lvDeviceStatusStruct, A LabView userevent data package
- typedef void(__cdecl * [SetLVUserEventDeviceInfoFuncPtr](#)) (unsigned long *lvUserEventRef)
- typedef struct [lvRegisterStatusStruct](#) [LabViewRegisterStatusType](#)
lvRegisterStatusStruct, A LabView userevent data package
- typedef void(__cdecl * [SetLVUserEventRegisterInfoFuncPtr](#)) (unsigned long *lvUserEventRef)
- [NKTPDLL_EXPORT](#) void [setLVUserEventPortInfo](#) (unsigned long *lvUserEventRef)
Enables/Disables labView user events for port status changes. Disable events by parsing in a zero value.
- [NKTPDLL_EXPORT](#) void [setLVUserEventDeviceInfo](#) (unsigned long *lvUserEventRef)
Enables/Disables labView user events for device status changes. Disable events by parsing in a zero value.
- [NKTPDLL_EXPORT](#) void [setLVUserEventRegisterInfo](#) (unsigned long *lvUserEventRef)
Enables/Disables labView user events for register status changes. Disable events by parsing in a zero value.

5.1.1 Detailed Description

NKTP DLL Interface, a communication DLL for interfacing to NKT Photonics products being controlled via the Interbus protocol. The NKTPDLL abstracts the burden of telegram creation and communication handling when communicating/controlling the NKT Photonics products.

Author

HCH

Date

23 June 2017

5.1.2 Macro Definition Documentation

5.1.2.1 NKTPDLL_EXPORT

```
#define NKTPDLL_EXPORT __declspec(dllimport)
```

5.1.3 Typedef Documentation

5.1.3.1 PortResultTypes

```
typedef unsigned char PortResultTypes
```

5.1.3.2 P2PPortResultTypes

```
typedef unsigned char P2PPortResultTypes
```

5.1.3.3 DeviceResultTypes

```
typedef unsigned char DeviceResultTypes
```

5.1.3.4 DeviceModeTypes

```
typedef unsigned char DeviceModeTypes
```

5.1.3.5 RegisterResultTypes

```
typedef unsigned char RegisterResultTypes
```

5.1.3.6 RegisterDataTypes

```
typedef unsigned char RegisterDataTypes
```

5.1.3.7 RegisterPriorityTypes

```
typedef unsigned char RegisterPriorityTypes
```

5.1.3.8 PortStatusTypes

```
typedef unsigned char PortStatusTypes
```

5.1.3.9 DeviceStatusTypes

```
typedef unsigned char DeviceStatusTypes
```

5.1.3.10 RegisterStatusTypes

```
typedef unsigned char RegisterStatusTypes
```

5.1.3.11 DateTimeType

```
typedef struct tDateTimeStruct DateTimeType
```

The tDateTime struct 24 hour format.

5.1.3.12 ParamSetUnitTypes

```
typedef unsigned char ParamSetUnitTypes
```

5.1.3.13 ParameterSetType

```
typedef struct tParamSetStruct ParameterSetType
```

The tParameterSet struct.

Note

This is how calculation on parametersets is done internally by modules:

DAC_value = (value * (X/Y)) + Offset; Where value is either [ParameterSetType::StartVal](#) or [ParameterSetType::FactoryVal](#)

value = (ADC_value * (X/Y)) + Offset; Where value often is available via another measurement register

5.1.3.14 GetAllPortsFuncPtr

```
typedef void(__cdecl * GetAllPortsFuncPtr) (char *portnames, unsigned short *maxLen)
```

5.1.3.15 GetOpenPortsFuncPtr

```
typedef void(__cdecl * GetOpenPortsFuncPtr) (char *portnames, unsigned short *maxLen)
```

5.1.3.16 PointToPointPortAddFuncPtr

```
typedef P2PPortResultTypes(__cdecl * PointToPointPortAddFuncPtr) (const char *portname, const char *hostAddress, const unsigned short hostPort, const char *clientAddress, const unsigned short clientPort, const unsigned char protocol, const unsigned char msTimeout)
```

5.1.3.17 PointToPointPortGetFuncPtr

```
typedef P2PPortResultTypes(__cdecl * PointToPointPortGetFuncPtr) (const char *portname, char *hostAddress, unsigned char *hostMaxLen, unsigned short *hostPort, char *clientAddress, unsigned char *clientMaxLen, unsigned short *clientPort, unsigned char *protocol, unsigned char *msTimeout)
```

5.1.3.18 PointToPointPortDelFuncPtr

```
typedef P2PPortResultTypes(__cdecl * PointToPointPortDelFuncPtr) (const char *portname)
```

5.1.3.19 OpenPortsFuncPtr

```
typedef PortResultTypes(__cdecl * OpenPortsFuncPtr) (const char *portnames, const char autoMode, const char liveMode)
```

5.1.3.20 ClosePortsFuncPtr

```
typedef PortResultTypes(__cdecl * ClosePortsFuncPtr) (const char *portnames)
```

5.1.3.21 SetLegacyBusScanningFuncPtr

```
typedef void(__cdecl * SetLegacyBusScanningFuncPtr) (const char legacyScanning)
```

5.1.3.22 GetLegacyBusScanningFuncPtr

```
typedef unsigned char(__cdecl * GetLegacyBusScanningFuncPtr) ()
```

5.1.3.23 SetSpecificBusScanningRangeFuncPtr

```
typedef void(__cdecl * SetSpecificBusScanningRangeFuncPtr) (const char specificScanning, const unsigned char startAddress, const unsigned char endAddress)
```

5.1.3.24 GetSpecificBusScanningRangeFuncPtr

```
typedef unsigned char(__cdecl * GetSpecificBusScanningRangeFuncPtr) (char *specificScanning, unsigned char *startAddress, unsigned char *endAddress)
```

5.1.3.25 getPortStatusFuncPtr

```
typedef PortResultTypes(__cdecl * getPortStatusFuncPtr) (const char *portname, PortStatusTypes *portStatus)
```

5.1.3.26 getPortErrorMsgFuncPtr

```
typedef PortResultTypes(__cdecl * getPortErrorMsgFuncPtr) (const char *portname, char *error↵ Message, unsigned short *maxLen)
```

5.1.3.27 RegisterReadFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadFuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, void *readData, unsigned char *readSize, const short index)
```

5.1.3.28 RegisterReadU8FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadU8FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned char *value, const short index)
```

5.1.3.29 RegisterReadS8FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadS8FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, signed char *value, const short index)
```

5.1.3.30 RegisterReadU16FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadU16FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned short *value, const short index)
```

5.1.3.31 RegisterReadS16FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadS16FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, signed short *value, const short index)
```

5.1.3.32 RegisterReadU32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadU32FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned long *value, const short index)
```

5.1.3.33 RegisterReadS32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadS32FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, signed long *value, const short index)
```

5.1.3.34 RegisterReadU64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadU64FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned long long *value, const short index)
```

5.1.3.35 RegisterReadS64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadS64FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, signed long long *value, const short index)
```

5.1.3.36 RegisterReadF32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadF32FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, float *value, const short index)
```

5.1.3.37 RegisterReadF64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadF64FuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, double *value, const short index)
```

5.1.3.38 RegisterReadAsciiFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterReadAsciiFuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, char *readStr, unsigned char *maxLen, const short index)
```


5.1.3.39 RegisterWriteFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteFuncPtr) (const char *portname, const unsigned
char devId, const unsigned char regId, const void *writeData, const unsigned char writeSize,
const short index)
```

5.1.3.40 RegisterWriteU8FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteU8FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned char value, const short index)
```

5.1.3.41 RegisterWriteS8FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteS8FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed char value, const short index)
```

5.1.3.42 RegisterWriteU16FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteU16FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned short value, const short index)
```

5.1.3.43 RegisterWriteS16FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteS16FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed short value, const short index)
```

5.1.3.44 RegisterWriteU32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteU32FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned long value, const short index)
```

5.1.3.45 RegisterWriteS32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteS32FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed long value, const short index)
```

5.1.3.46 RegisterWriteU64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteU64FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned long long value, const short
index)
```

5.1.3.47 RegisterWriteS64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteS64FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed long long value, const short
index)
```

5.1.3.48 RegisterWriteF32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteF32FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const float value, const short index)
```

5.1.3.49 RegisterWriteF64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteF64FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const double value, const short index)
```

5.1.3.50 RegisterWriteAsciiFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteAsciiFuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const char *writeStr, const char writeEOL,
const short index)
```

5.1.3.51 RegisterWriteReadFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadFuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const void *writeData, const unsigned char
writeSize, void *readData, unsigned char *readSize, const short index)
```

5.1.3.52 RegisterWriteReadU8FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadU8FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned char writeValue, unsigned char
*readValue, const short index)
```

5.1.3.53 RegisterWriteReadS8FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadS8FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed char writeValue, signed char
*readValue, const short index)
```

5.1.3.54 RegisterWriteReadU16FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadU16FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned short writeValue, unsigned
short *readValue, const short index)
```

5.1.3.55 RegisterWriteReadS16FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadS16FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed short writeValue, signed short
*readValue, const short index)
```

5.1.3.56 RegisterWriteReadU32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadU32FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned long writeValue, unsigned long
*readValue, const short index)
```

5.1.3.57 RegisterWriteReadS32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadS32FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed long writeValue, signed long
*readValue, const short index)
```

5.1.3.58 RegisterWriteReadU64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadU64FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const unsigned long long writeValue, unsigned
long long *readValue, const short index)
```

5.1.3.59 RegisterWriteReadS64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadS64FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const signed long long writeValue, signed long
long *readValue, const short index)
```

5.1.3.60 RegisterWriteReadF32FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadF32FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const float writeValue, float *readValue,
const short index)
```

5.1.3.61 RegisterWriteReadF64FuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadF64FuncPtr) (const char *portname, const
unsigned char devId, const unsigned char regId, const double writeValue, double *readValue,
const short index)
```

5.1.3.62 RegisterWriteReadAsciiFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterWriteReadAsciiFuncPtr) (const char *portname,
const unsigned char devId, const unsigned char regId, const char *writeStr, const char write↵
EOL, char *readStr, unsigned char *maxLen, const short index)
```

5.1.3.63 DeviceGetTypeFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetTypeFuncPtr) (const char *portname, const unsigned char devId, unsigned char *devType)
```

5.1.3.64 DeviceGetTypeV2FuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetTypeV2FuncPtr) (const char *portname, const unsigned char devId, unsigned short *devType)
```

5.1.3.65 DeviceGetSysTypeFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetSysTypeFuncPtr) (const char *portname, const unsigned char devId, unsigned char *sysType)
```

5.1.3.66 DeviceGetPartNumberStrFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetPartNumberStrFuncPtr) (const char *portname, const unsigned char devId, char *partnumber, unsigned char *maxLen)
```

5.1.3.67 DeviceGetPCBVersionFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetPCBVersionFuncPtr) (const char *portname, const unsigned char devId, unsigned char *PCBVersion)
```

5.1.3.68 DeviceGetStatusBitsFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetStatusBitsFuncPtr) (const char *portname, const unsigned char devId, unsigned long *statusBits)
```

5.1.3.69 DeviceGetErrorCodeFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetErrorCodeFuncPtr) (const char *portname, const unsigned char devId, unsigned short *errorCode)
```

5.1.3.70 DeviceGetBootloaderVersionFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetBootloaderVersionFuncPtr) (const char *portname, const unsigned char devId, unsigned short *version)
```

5.1.3.71 DeviceGetBootloaderVersionStrFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetBootloaderVersionStrFuncPtr) (const char *portname, const unsigned char devId, char *versionStr, unsigned char *maxLen)
```

5.1.3.72 DeviceGetFirmwareVersionFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetFirmwareVersionFuncPtr) (const char *portname,  
const unsigned char devId, unsigned short *version)
```

5.1.3.73 DeviceGetFirmwareVersionStrFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetFirmwareVersionStrFuncPtr) (const char *portname,  
const unsigned char devId, char *versionStr, unsigned char *maxLen)
```

5.1.3.74 DeviceGetModuleSerialNumberStrFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetModuleSerialNumberStrFuncPtr) (const char *portname,  
const unsigned char devId, char *serialNumber, unsigned char *maxLen)
```

5.1.3.75 DeviceGetPCBSerialNumberStrFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetPCBSerialNumberStrFuncPtr) (const char *portname,  
const unsigned char devId, char *serialNumber, unsigned char *maxLen)
```

5.1.3.76 DeviceCreateFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceCreateFuncPtr) (const char *portname, const unsigned  
char devId, const char waitReady)
```

5.1.3.77 DeviceExistsFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceExistsFuncPtr) (const char *portname, const unsigned  
char devId, unsigned char *exists)
```

5.1.3.78 DeviceRemoveFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceRemoveFuncPtr) (const char *portname, const unsigned  
char devId)
```

5.1.3.79 DeviceRemoveAllFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceRemoveAllFuncPtr) (const char *portname)
```

5.1.3.80 DeviceGetAllTypesFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetAllTypesFuncPtr) (const char *portname, unsigned  
char *types, unsigned char *maxTypes)
```

5.1.3.81 DeviceGetAllTypesV2FuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetAllTypesV2FuncPtr) (const char *portname, unsigned short *types, unsigned char *maxTypes)
```

5.1.3.82 DeviceGetModeFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetModeFuncPtr) (const char *portname, const unsigned char devId, unsigned char *devMode)
```

5.1.3.83 DeviceGetLiveFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceGetLiveFuncPtr) (const char *portname, const unsigned char devId, unsigned char *liveMode)
```

5.1.3.84 DeviceSetLiveFuncPtr

```
typedef DeviceResultTypes(__cdecl * DeviceSetLiveFuncPtr) (const char *portname, const unsigned char devId, const unsigned char liveMode)
```

5.1.3.85 RegisterCreateFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterCreateFuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, const RegisterPriorityTypes priority, const RegisterDataTypes dataType)
```

5.1.3.86 RegisterExistsFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterExistsFuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId, unsigned char *exists)
```

5.1.3.87 RegisterRemoveFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterRemoveFuncPtr) (const char *portname, const unsigned char devId, const unsigned char regId)
```

5.1.3.88 RegisterRemoveAllFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterRemoveAllFuncPtr) (const char *portname, const unsigned char devId)
```

5.1.3.89 RegisterGetAllFuncPtr

```
typedef RegisterResultTypes(__cdecl * RegisterGetAllFuncPtr) (const char *portname, const unsigned char devId, unsigned char *regs, unsigned char *maxRegs)
```

5.1.3.90 PortStatusCallbackFuncPtr

```
typedef void(__cdecl * PortStatusCallbackFuncPtr) (const char *portname, const PortStatusTypes status, const unsigned char curScanAdr, const unsigned char maxScanAdr, const unsigned short foundType)
```

Defines the PortStatusCallbackFuncPtr for the [openPorts](#) and [closePorts](#) functions.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the current portname. |
| <i>status</i> | The current port status as a PortStatusTypes with a tPortStatusTypes value. |
| <i>curScanAdr</i> | When status is PortScanProgress or PortScanDeviceFound this indicates the current module address scanned or found. |
| <i>maxScanAdr</i> | When status is PortScanProgress or PortScanDeviceFound this indicates the last module address to be scanned. |
| <i>foundType</i> | When status is PortScanDeviceFound this value will represent the found module type. |

Note

Was originally a byte, but since we now has moduletypes above 0xFF, this has been changed to a short (unsigned 16bit).

Please note that due to risk of circular runaway leading to stack overflow, it is not allowed to call functions in the DLL from within the callback function. If a call is made to a function in the DLL the function will therefore return an application busy error.

5.1.3.91 SetCallbackPtrPortInfoFuncPtr

```
typedef void(__cdecl * SetCallbackPtrPortInfoFuncPtr) (PortStatusCallbackFuncPtr callback)
```

5.1.3.92 DeviceStatusCallbackFuncPtr

```
typedef void(__cdecl * DeviceStatusCallbackFuncPtr) (const char *portname, const unsigned char devId, const DeviceStatusTypes status, const unsigned char devDataLen, const void *devData)
```

Defines the DeviceStatusCallbackFuncPtr for the devices created with the [deviceCreate](#) function or created automatically via the [openPorts](#) function (Having autoMode = 1).

Parameters

| | |
|-------------------|---|
| <i>portname</i> | Zero terminated string giving the current portname. |
| <i>devId</i> | The device id (module address). |
| <i>status</i> | The current port status as a DeviceStatusTypes with a tDeviceStatusTypes value. |
| <i>devDataLen</i> | Number of bytes in devData. |
| <i>devData</i> | The device data as specified in tDeviceStatusTypes status. |

Note

Please note that due to risk of circular runaway leading to stack overflow, it is not allowed to call functions in the DLL from within the callback function. If a call is made to a function in the DLL the function will therefore return an application busy error.

5.1.3.93 SetCallbackPtrDeviceInfoFuncPtr

```
typedef void(__cdecl * SetCallbackPtrDeviceInfoFuncPtr) (DeviceStatusCallbackFuncPtr callback)
```

5.1.3.94 RegisterStatusCallbackFuncPtr

```
typedef void(__cdecl * RegisterStatusCallbackFuncPtr) (const char *portname, const unsigned
char devId, const unsigned char regId, const RegisterStatusTypes status, const RegisterDataTypes
regType, const unsigned char regDataLen, const void *regData)
```

Defines the RegisterStatusCallbackFuncPtr for the registers created or connected with the [registerCreate](#) function.

Parameters

| | |
|-------------------|---|
| <i>portname</i> | Zero terminated string giving the current portname. |
| <i>devId</i> | The Id/Address of the device. |
| <i>regId</i> | The register number. |
| <i>status</i> | The current register status as a RegisterStatusTypes with a tRegisterStatusTypes value. |
| <i>regType</i> | The RegisterDataTypes , not used internally but could be used in a common callback function to determine data type. |
| <i>regDataLen</i> | Number of databytes. |
| <i>regData</i> | The register data. |

Note

Please note that due to risk of circular runaway leading to stack overflow, it is not allowed to call functions in the DLL from within the callback function. If a call is made to a function in the DLL the function will therefore return an application busy error.

5.1.3.95 SetCallbackPtrRegisterInfoFuncPtr

```
typedef void(__cdecl * SetCallbackPtrRegisterInfoFuncPtr) (RegisterStatusCallbackFuncPtr callback)
```

5.1.3.96 LabViewPortStatusType

```
typedef struct lvPortStatusStruct LabViewPortStatusType
```

[lvPortStatusStruct](#), A LabView userevent data package

5.1.3.97 SetLVUserEventPortInfoFuncPtr

```
typedef void(__cdecl * SetLVUserEventPortInfoFuncPtr) (unsigned long *lvUserEventRef)
```

5.1.3.98 LabViewDeviceStatusType

```
typedef struct lvDeviceStatusStruct LabViewDeviceStatusType
```

[lvDeviceStatusStruct](#), A LabView userevent data package

5.1.3.99 SetLVUserEventDeviceInfoFuncPtr

```
typedef void(__cdecl * SetLVUserEventDeviceInfoFuncPtr) (unsigned long *lvUserEventRef)
```

5.1.3.100 LabViewRegisterStatusType

```
typedef struct lvRegisterStatusStruct LabViewRegisterStatusType
```

[lvRegisterStatusStruct](#), A LabView userevent data package

5.1.3.101 SetLVUserEventRegisterInfoFuncPtr

```
typedef void(__cdecl * SetLVUserEventRegisterInfoFuncPtr) (unsigned long *lvUserEventRef)
```

5.1.4 Enumeration Type Documentation

5.1.4.1 tPortResultTypes

```
enum tPortResultTypes
```

The tPortResultTypes enum.

Enumerator

| | |
|-------------------|--|
| OPSuccess | 0 - Successfull operation. |
| OPFailed | 1 - The openPorts function has failed. |
| OPPortNotFound | 2 - The specified portname could not be found. |
| OPNoDevices | 3 - No devices found on the specified port. |
| OPApplicationBusy | 4 - The function is not allowed to be invoked from within a callback function. |

5.1.4.2 tP2PPortResultTypes

```
enum tP2PPortResultTypes
```

The tPointToPointPortStatus enum.

Enumerator

| | |
|---------------------|--|
| P2PSuccess | 0 - Successfull operation. |
| P2PInvalidPortname | 1 - Invalid portname provided. |
| P2PInvalidLocalIP | 2 - Invalid local IP provided. |
| P2PInvalidRemoteIP | 3 - Invalid remote IP provided. |
| P2PPortnameNotFound | 4 - Portname not found. |
| P2PPortnameExists | 5 - Portname already exists. |
| P2PApplicationBusy | 6 - The function is not allowed to be invoked from within a callback function. |

5.1.4.3 tDeviceResultTypes

```
enum tDeviceResultTypes
```

The tDeviceResultTypes enum.

Enumerator

| | |
|--------------------------|--|
| DevResultSuccess | 0 - Successfull operation. |
| DevResultWaitTimeout | 1 - The function deviceCreate , timed out waiting for the device being ready. |
| DevResultFailed | 2 - The function deviceCreate , failed. |
| DevResultDeviceNotFound | 3 - The specified device could not be found in the internal device list. |
| DevResultPortNotFound | 4 - The function deviceCreate , failed due to not being able to find the specified port. |
| DevResultPortOpenError | 5 - The function deviceCreate , failed due to port not being open. |
| DevResultApplicationBusy | 6 - The function is not allowed to be invoked from within a callback function. |

5.1.4.4 tDeviceModeTypes

```
enum tDeviceModeTypes
```

The tDeviceModeTypes enum.

Enumerator

| | |
|--------------------|--|
| DevModeDisabled | 0 - The device is disabled. Not being polled and serviced. |
| DevModeAnalyzeInit | 1 - The analyze cycle has been started for the device. |
| DevModeAnalyze | 2 - The analyze cycle is in progress. All default registers being read to determine its state. |
| DevModeNormal | 3 - The analyze cycle has completed and the device is ready. |
| DevModeLogDownload | 4 - A log is being downloaded from the device. |
| DevModeError | 5 - The device is in an error state. Deprecated No longer used. |
| DevModeTimeout | 6 - The connection to the device has been lost. |
| DevModeUpload | 7 - The device is in upload mode and can not be used normally. |

5.1.4.5 tRegisterResultTypes

```
enum tRegisterResultTypes
```

The tRegisterResultTypes enum.

Enumerator

| | |
|--------------------|---|
| RegResultSuccess | 0 - Successfull operation. |
| RegResultReadError | 1 - Arises from a registerWrite function with index > 0, if the pre-read fails. |
| RegResultFailed | 2 - The function registerCreate has failed. |

Enumerator

| | |
|---------------------------|--|
| RegResultBusy | 3 - The module has reported a BUSY error, the kernel automatically retries on busy but have given up. |
| RegResultNacked | 4 - The module has Nacked the register, which typically means non existing register. |
| RegResultCRCError | 5 - The module has reported a CRC error, which means the received message has CRC errors. |
| RegResultTimeout | 6 - The module has not responded in time. A module should respond in max. 75ms |
| RegResultComError | 7 - The module has reported a COM error, which typically means out of sync or garbage error. |
| RegResultTypeError | 8 - The datatype does not seem to match the register datatype. |
| RegResultIndexError | 9 - The index seem to be out of range of the register length. |
| RegResultPortClosed | 10 - The specified port is closed error. Could happen if the USB is unplugged in the middle of a sequence. |
| RegResultRegisterNotFound | 11 - The specified register could not be found in the internal register list for the specified device. |
| RegResultDeviceNotFound | 12 - The specified device could not be found in the internal device list. |
| RegResultPortNotFound | 13 - The specified portname could not be found. |
| RegResultPortOpenError | 14 - The specified portname could not be opened. The port might be in use by another application. |
| RegResultApplicationBusy | 15 - The function is not allowed to be invoked from within a callback function. |

5.1.4.6 tRegisterDataTypes

```
enum tRegisterDataTypes
```

The tRegisterDataTypes enum.

Enumerator

| | |
|------------------|---|
| RegData_Unknown | 0 - Unknown/Undefined data type. |
| RegData_Mixed | 1 - Mixed content data type. |
| RegData_U8 | 2 - 8 bit unsigned data type (unsigned char). |
| RegData_S8 | 3 - 8 bit signed data type (char). |
| RegData_U16 | 4 - 16 bit unsigned data type (unsigned short). |
| RegData_S16 | 5 - 16 bit signed data type (short). |
| RegData_U32 | 6 - 32 bit unsigned data type (unsigned long). |
| RegData_S32 | 7 - 32 bit signed data type (long). |
| RegData_F32 | 8 - 32 bit floating point data type (float). |
| RegData_U64 | 9 - 64 bit unsigned data type (unsigned long long). |
| RegData_S64 | 10 - 64 bit signed data type (long long). |
| RegData_F64 | 11 - 64 bit floating point data type (double). |
| RegData_Ascii | 12 - Zero terminated ascii string data type. |
| RegData_Paramset | 13 - Parameterset data type. ParameterSetType |
| RegData_B8 | 14 - 8 bit binary data type (unsigned char). |
| RegData_H8 | 15 - 8 bit hexadecimal data type (unsigned char). |
| RegData_B16 | 16 - 16 bit binary data type (unsigned short). |

Enumerator

| | |
|------------------|---|
| RegData_H16 | 17 - 16 bit hexadecimal data type (unsigned short). |
| RegData_B32 | 18 - 32 bit binary data type (unsigned long). |
| RegData_H32 | 19 - 32 bit hexadecimal data type (unsigned long). |
| RegData_B64 | 20 - 64 bit binary data type (unsigned long long). |
| RegData_H64 | 21 - 64 bit hexadecimal data type (unsigned long long). |
| RegData_DateTime | 22 - Datetime data type. DateTimeType |

5.1.4.7 tRegisterPriorityTypes

```
enum tRegisterPriorityTypes
```

The tRegisterPriorityTypes enum.

Enumerator

| | |
|------------------|--|
| RegPriority_Low | 0 - The register is polled with low priority. |
| RegPriority_High | 1 - The register is polled with high priority. |

5.1.4.8 tPortStatusTypes

```
enum tPortStatusTypes
```

The tPortStatusTypes enum.

Enumerator

| | |
|---------------------|-----------------------------------|
| PortStatusUnknown | 0 - Unknown status. |
| PortOpening | 1 - The port is opening. |
| PortOpened | 2 - The port is now open. |
| PortOpenFail | 3 - The port open failed. |
| PortScanStarted | 4 - The port scanning is started. |
| PortScanProgress | 5 - The port scanning progress. |
| PortScanDeviceFound | 6 - The port scan found a device. |
| PortScanEnded | 7 - The port scanning ended. |
| PortClosing | 8 - The port is closing. |
| PortClosed | 9 - The port is now closed. |
| PortReady | 10 - The port is open and ready. |

5.1.4.9 tDeviceStatusTypes

```
enum tDeviceStatusTypes
```

The tDeviceStatusTypes enum.

Enumerator

| | |
|---------------------------|---|
| DeviceModeChanged | 0 - devData contains 1 unsigned byte DeviceModeTypes |
| DeviceLiveChanged | 1 - devData contains 1 unsigned byte, 0=live off, 1=live on. |
| DeviceTypeChanged | 2 - devData contains 1 unsigned short with DeviceType (module type). Note Was originally a byte, but since we now has moduletypes above 0xFF, this has been changed to a short(unsigned 16bit). |
| DevicePartNumberChanged | 3 - devData contains a zero terminated string with partnumber. Deprecated No longer available. |
| DevicePCBVersionChanged | 4 - devData contains 1 unsigned byte with PCB version number. |
| DeviceStatusBitsChanged | 5 - devData contains 1 unsigned long with statusbits. |
| DeviceErrorCodeChanged | 6 - devData contains 1 unsigned short with errorcode. |
| DeviceBIVerChanged | 7 - devData contains a zero terminated string with Bootloader version. |
| DeviceFwVerChanged | 8 - devData contains a zero terminated string with Firmware version. |
| DeviceModuleSerialChanged | 9 - devData contains a zero terminated string with Module serialnumber. |
| DevicePCBSerialChanged | 10 - devData contains a zero terminated string with PCB serialnumber. |
| DeviceSysTypeChanged | 11 - devData contains 1 unsigned byte with SystemType (system type). |

5.1.4.10 tRegisterStatusTypes

```
enum tRegisterStatusTypes
```

The tRegisterStatusTypes enum.

Enumerator

| | |
|-------------|--|
| RegSuccess | 0 - Register operation was successfull. |
| RegBusy | 1 - Register operation resulted in a busy. |
| RegNacked | 2 - Register operation resulted in a nack, seems to be non existing register. |
| RegCRCErr | 3 - Register operation resulted in a CRC error. |
| RegTimeout | 4 - Register operation resulted in a timeout. |
| RegComError | 5 - Register operation resulted in a COM error. Out of sync. or garbage error. |

5.1.4.11 tParamSetUnitTypes

```
enum tParamSetUnitTypes
```

The tParamSetUnitTypes enum.

Enumerator

| | |
|----------|------------------|
| UnitNone | 0 - none/unknown |
| UnitmV | 1 - mV |
| UnitV | 2 - V |

Enumerator

| | |
|--------------|------------------------|
| UnituA | 3 - μ A |
| UnitmA | 4 - mA |
| UnitA | 5 - A |
| UnituW | 6 - μ W |
| UnitcmW | 7 - mW/100 |
| UnitdmW | 8 - mW/10 |
| UnitmW | 9 - mW |
| UnitW | 10 - W |
| UnitmC | 11 - $^{\circ}$ C/1000 |
| UnitcC | 12 - $^{\circ}$ C/100 |
| UnitdC | 13 - $^{\circ}$ C/10 |
| Unitpm | 14 - pm |
| Unitdnm | 15 - nm/10 |
| Unitnm | 16 - nm |
| UnitPerCent | 17 - % |
| UnitPerMille | 18 - ‰ |
| UnitcmA | 19 - mA/100 |
| UnitdmA | 20 - mA/10 |
| UnitRPM | 21 - RPM |
| UnitdBm | 22 - dBm |
| UnitcBm | 23 - dBm/10 |
| UnitmBm | 24 - dBm/100 |
| UnitdB | 25 - dB |
| UnitcB | 26 - dB/10 |
| UnitmB | 27 - dB/100 |
| Unitdpm | 28 - pm/10 |
| UnitcV | 29 - V/100 |
| UnitdV | 30 - V/10 |
| Unitlm | 31 - lm (lumen) |
| Unitdlm | 32 - lm/10 |
| Unitclm | 33 - lm/100 |
| Unitmlm | 34 - lm/1000 |
| UnitHz | 35 - Hz |
| UnitkHz | 36 - kHz |
| UnitMHz | 37 - MHz |
| UnitSec | 38 - s |
| UnitmSec | 39 - ms |
| UnituSec | 40 - μ s |
| UnitdA | 41 - A/10 |
| UnitcA | 42 - A/100 |
| UnitduA | 43 - μ A/10 |
| UnitcuA | 44 - μ A/100 |
| UnitnA | 45 - nA |
| UnitdW | 46 - W/10 |
| UnitcW | 47 - W/100 |
| UnitpA | 48 - pA |

5.1.5 Function Documentation

5.1.5.1 getAllPorts()

```
NKTPDLL_EXPORT void getAllPorts (
    char * portnames,
    unsigned short * maxLen )
```

Returns a comma separated string with all existing ports.

Parameters

| | |
|------------------|--|
| <i>portnames</i> | Pointer to a preallocated string area where the function will store the comma separated string. |
| <i>maxLen</i> | Size of preallocated string area. The returned string may be truncated to fit into the allocated area. |

5.1.5.2 getOpenPorts()

```
NKTPDLL_EXPORT void getOpenPorts (
    char * portnames,
    unsigned short * maxLen )
```

Returns a comma separated string with all already opened ports.

Parameters

| | |
|------------------|--|
| <i>portnames</i> | Pointer to a preallocated string area where the function will store the comma separated string. |
| <i>maxLen</i> | Size of preallocated string area. The returned string may be truncated to fit into the allocated area. |

5.1.5.3 pointToPointPortAdd()

```
NKTPDLL_EXPORT P2PPortResultTypes pointToPointPortAdd (
    const char * portname,
    const char * hostAddress,
    const unsigned short hostPort,
    const char * clientAddress,
    const unsigned short clientPort,
    const unsigned char protocol,
    const unsigned char msTimeout )
```

Creates or Modifies a point to point port.

Parameters

| | |
|----------------------|--|
| <i>portname</i> | Zero terminated string giving the portname. ex. "AcoustikPort1" |
| <i>hostAddress</i> | Zero terminated string giving the local ip address. ex. "192.168.1.67" |
| <i>hostPort</i> | The local port number. |
| <i>clientAddress</i> | Zero terminated string giving the remote ip address. ex. "192.168.1.100" |
| <i>clientPort</i> | The remote port number. |

Parameters

| | |
|------------------|--|
| <i>protocol</i> | <ul style="list-style-type: none"> • 0 Specifies TCP protocol. • 1 Specifies UDP protocol. |
| <i>msTimeout</i> | Telegram timeout value in milliseconds, typically set to 100ms. |

Returns

[tP2PPortResultTypes](#)

5.1.5.4 pointToPointPortGet()

```
NKTPDLL_EXPORT P2PPortResultTypes pointToPointPortGet (
    const char * portname,
    char * hostAddress,
    unsigned char * hostMaxLen,
    unsigned short * hostPort,
    char * clientAddress,
    unsigned char * clientMaxLen,
    unsigned short * clientPort,
    unsigned char * protocol,
    unsigned char * msTimeout )
```

Retrieve an already created point to point port setting.

Parameters

| | |
|----------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). ex. "AcoustikPort1" |
| <i>hostAddress</i> | Pointer to a preallocated string area where the function will store the zero terminated string, describing the local ip address. |
| <i>hostMaxLen</i> | Pointer to an unsigned char giving the size of the preallocated hostAddress area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |
| <i>hostPort</i> | Pointer to a preallocated unsigned short where the function will store the local port number. |
| <i>clientAddress</i> | Pointer to a preallocated string area where the function will store the zero terminated string, describing the remote ip address. |
| <i>clientMaxLen</i> | Pointer to an unsigned char giving the size of the preallocated clientAddress area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |
| <i>clientPort</i> | Pointer to a preallocated unsigned short where the function will store the client port number. |
| <i>protocol</i> | Pointer to a preallocated char where the function will store the protocol. <ul style="list-style-type: none"> • 0 Specifies TCP protocol. • 1 Specifies UDP protocol. |
| <i>msTimeout</i> | Pointer to a preallocated char where the function will store the timeout value. |

Returns

[tP2PPortResultTypes](#)

5.1.5.5 pointToPointPortDel()

```
NKTPDLL_EXPORT P2PPortResultTypes pointToPointPortDel (
    const char * portname )
```

Delete an already created point to point port.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). ex. "AcoustikPort1" |
|-----------------|--|

Returns

[tP2PPortResultTypes](#)

5.1.5.6 openPorts()

```
NKTPDLL_EXPORT PortResultTypes openPorts (
    const char * portnames,
    const char autoMode,
    const char liveMode )
```

Opens the provided portname(s), or all available ports if an empty string provided. Repeatedly calls is allowed to reopen and/or rescan for devices.

Parameters

| | |
|------------------|---|
| <i>portnames</i> | Zero terminated comma separated string giving the portnames to open (case sensitive). An empty string opens all available ports. |
| <i>autoMode</i> | <ul style="list-style-type: none"> • 0 the openPorts function only opens the port. Busscanning and device creation is NOT automatically handled. • 1 the openPorts function will automatically start the busscanning and create the found devices in the internal devicelist. The port is automatically closed if no devices found. |
| <i>liveMode</i> | <ul style="list-style-type: none"> • 0 the openPorts function disables the continuously monitoring of the registers. No callback possible on register changes. Use registerRead, registerWrite & registerWriteRead functions. • 1 the openPorts function will keep all the found or created devices in live mode, which means the Interbus kernel keeps monitoring all the found devices and their registers. Please note that this will keep the modules watchdog alive as long as the port is open. |

Returns

[tPortResultTypes](#)

Note

The function may timeout after 2 seconds waiting for port ready status and return [OPFailed](#).
In case autoMode is specified this timeout is extended to 20 seconds to allow for busscanning to complete.

5.1.5.7 closePorts()

```
NKTPDLL_EXPORT PortResultTypes closePorts (
    const char * portnames )
```

Closes the provided portname(s), or all opened ports if an empty string provided.

Parameters

| | |
|------------------|---|
| <i>portnames</i> | Zero terminated comma separated string giving the portnames to close (case sensitive). An empty string closes all open ports. |
|------------------|---|

Returns

[tPortResultTypes](#)

Note

The function may timeout after 2 seconds waiting for port close to complete and return [OPFailed](#).

5.1.5.8 setLegacyBusScanning()

```
NKTPDLL_EXPORT void setLegacyBusScanning (
    const char legacyScanning )
```

Sets legacy busscanning on or off.

Parameters

| | |
|-----------------------|---|
| <i>legacyScanning</i> | <ul style="list-style-type: none"> • 0 the busscanning is set to normal mode and allows for rolling masterId. In this mode the masterId is changed for each message to allow for out of sync. detection. • 1 the busscanning is set to legacy mode and fixes the masterId at address 66(0x42). Some older/legacy modules does not accept masterIds other than 66(0x42). |
|-----------------------|---|

See also

[getLegacyBusScanning](#), [setSpecificBusScanningRange](#) and [getSpecificBusScanningRange](#)

Note

The buscanning can be defined with [setLegacyBusScanning](#) and [setSpecificBusScanningRange](#) according to following settings:

| SpecificBusScanning | LegacyBusScanning | Scanning procedure |
|---------------------|-------------------|--|
| 0 | 0 | If no module found at address 128 scans 1-40, if module found at address 128 scans 1-160 (This is the default) |
| 0 | 1 | Scans 1-40 with a fixed masterId 66(0x42) |
| 1 | 0 | Scans the specified range set with setSpecificBusScanningRange |
| 1 | 1 | Scans the specified range set with setSpecificBusScanningRange using a fixed masterId 66(0x42) |

5.1.5.9 getLegacyBusScanning()

```
NKTPDLL_EXPORT unsigned char getLegacyBusScanning ( )
```

Gets legacy busscanning status.

Returns

An unsigned char, with legacyScanning status. 0 the busscanning is currently in normal mode. 1 the busscanning is currently in legacy mode.

See also

[setLegacyBusScanning](#), [setSpecificBusScanningRange](#) and [getSpecificBusScanningRange](#)

5.1.5.10 setSpecificBusScanningRange()

```
NKTPDLL_EXPORT void setSpecificBusScanningRange (
    const char specificScanning,
    const unsigned char startAddress,
    const unsigned char endAddress )
```

Sets specific busscanning address range, on/off.

Parameters

| | |
|-------------------------|---|
| <i>specificScanning</i> | <ul style="list-style-type: none"> • 0 the specific busscanning range is set to Off (Normal mode). • 1 the specific busscanning range is set to On. And the range is set to the parameters startAddress and endAddress. |
| <i>startAddress</i> | The specific start address. |
| <i>endAddress</i> | The specific end address. |

See also

[getSpecificBusScanningRange](#), [setLegacyBusScanning](#) and [getLegacyBusScanning](#)

5.1.5.11 getSpecificBusScanningRange()

```
NKTPDLL_EXPORT void getSpecificBusScanningRange (
    char * specificScanning,
    unsigned char * startAddress,
    unsigned char * endAddress )
```

Gets specific busscanning address range and status.

Parameters

| | |
|-------------------------|---|
| <i>specificScanning</i> | Pointer to a char where the function will store the current status. 0 = Off, 1 = On |
| <i>startAddress</i> | Pointer to an unsigned char where the function will store the current startAddress. |
| <i>endAddress</i> | Pointer to an unsigned char where the function will store the current endAddress. |

See also

[setSpecificBusScanningRange](#), [setLegacyBusScanning](#) and [getLegacyBusScanning](#)

5.1.5.12 getPortStatus()

```
NKTPDLL_EXPORT PortResultTypes getPortStatus (
    const char * portname,
    PortStatusTypes * portStatus )
```

Retrieve [tPortStatusTypes](#) for a given port.

Parameters

| | |
|-------------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). ex. "COM1" |
| <i>portStatus</i> | Pointer to a PortStatusTypes where the function will store the port status. |

Returns

[tPortResultTypes](#)

5.1.5.13 getPortErrorMsg()

```
NKTPDLL_EXPORT PortResultTypes getPortErrorMsg (
    const char * portname,
    char * errorMessage,
    unsigned short * maxLen )
```

Retrieve error message for a given port. An empty string indicates no error.

Parameters

| | |
|---------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). ex. "COM1" |
| <i>errorMessage</i> | Pointer to a preallocated string area where the function will store the zero terminated error string. |
| <i>maxLen</i> | Pointer to an unsigned short giving the size of the preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |

Returns

[tPortResultTypes](#)

5.1.5.14 registerRead()

```
NKTPDLL_EXPORT RegisterResultTypes registerRead (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    void * readData,
    unsigned char * readSize,
    const short index )
```

Reads a register value and returns the result in readData area.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>readData</i> | Pointer to a preallocated data area where the function will store the register value. |
| <i>readSize</i> | Size of preallocated data area, modified by the function to reflect the actual length of the returned register value. The returned register value may be truncated to fit into the allocated area. |
| <i>index</i> | Data index. Typically -1, but could be used to extract data from a specific position in the register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

See also

[registerReadU8](#), [registerReadS8](#) etc.

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.15 registerReadU8()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadU8 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    unsigned char * value,
    const short index )
```

Reads an unsigned char (8bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to an unsigned char where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.16 registerReadS8()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadS8 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    signed char * value,
    const short index )
```

Reads a signed char (8bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to a signed char where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.17 registerReadU16()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadU16 (  
    const char * portname,  
    const unsigned char devId,  
    const unsigned char regId,  
    unsigned short * value,  
    const short index )
```

Reads an unsigned short (16bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to an unsigned short where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.18 registerReadS16()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadS16 (  
    const char * portname,  
    const unsigned char devId,  
    const unsigned char regId,  
    signed short * value,  
    const short index )
```

Reads a signed short (16bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to a signed short where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.19 registerReadU32()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadU32 (  
    const char * portname,  
    const unsigned char devId,  
    const unsigned char regId,  
    unsigned long * value,  
    const short index )
```

Reads an unsigned long (32bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to an unsigned long where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.20 registerReadS32()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadS32 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    signed long * value,
    const short index )
```

Reads a signed long (32bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to a signed long where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.21 registerReadU64()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadU64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    unsigned long long * value,
    const short index )
```

Reads an unsigned long long (64bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to an unsigned long long where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.22 registerReadS64()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadS64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    signed long long * value,
    const short index )
```

Reads a signed long long (64bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to a signed long long where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.23 registerReadF32()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadF32 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    float * value,
    const short index )
```

Reads a float (32bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to a float where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.24 registerReadF64()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadF64 (  
    const char * portname,  
    const unsigned char devId,  
    const unsigned char regId,  
    double * value,  
    const short index )
```

Reads a double (64bit) register value and returns the result in value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | Pointer to a double where the function will store the register value. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a value in a multi value register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.25 registerReadAscii()

```
NKTPDLL_EXPORT RegisterResultTypes registerReadAscii (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    char * readStr,
    unsigned char * maxLen,
    const short index )
```

Reads a Ascii string register value and returns the result in readStr area.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>readStr</i> | Pointer to a preallocated string area where the function will store the register value. |
| <i>maxLen</i> | Size of preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |
| <i>index</i> | Value index. Typically -1, but could be used to extract a string in a mixed type register. Index is byte counted. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.26 registerWrite()

```
NKTPDLL_EXPORT RegisterResultTypes registerWrite (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const void * writeData,
    const unsigned char writeSize,
    const short index )
```

Writes a register value.

Parameters

| | |
|------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeData</i> | Pointer to a data area from where the write value will be extracted. |
| <i>writeSize</i> | Size of data area, ex. number of bytes to write. Write size is limited to max 240 bytes |
| <i>index</i> | Data index. Typically -1, but could be used to write data at a specific position in the register. Index is byte counted. Index >= 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

See also

[registerWriteU8](#), [registerWriteS8](#) etc.

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.27 registerWriteU8()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteU8 (  
    const char * portname,  
    const unsigned char devId,  
    const unsigned char regId,  
    const unsigned char value,  
    const short index )
```

Writes an unsigned char (8bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.28 registerWriteS8()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteS8 (  
    const char * portname,  
    const unsigned char devId,
```

```

    const unsigned char regId,
    const signed char value,
    const short index )

```

Writes a signed char (8bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.29 registerWriteU16()

```

NKTPDLL_EXPORT RegisterResultTypes registerWriteU16 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const unsigned short value,
    const short index )

```

Writes an unsigned short (16bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.30 registerWriteS16()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteS16 (  
    const char * portname,  
    const unsigned char devId,  
    const unsigned char regId,  
    const signed short value,  
    const short index )
```

Writes a signed short (16bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.31 registerWriteU32()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteU32 (  
    const char * portname,  
    const unsigned char devId,  
    const unsigned char regId,  
    const unsigned long value,  
    const short index )
```

Writes an unsigned long (32bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.32 registerWriteS32()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteS32 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const signed long value,
    const short index )
```

Writes a signed long (32bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.33 registerWriteU64()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteU64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const unsigned long long value,
    const short index )
```

Writes an unsigned long long (64bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.34 registerWriteS64()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteS64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const signed long long value,
    const short index )
```

Writes a signed long long (64bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.35 registerWriteF32()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteF32 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const float value,
    const short index )
```

Writes a float (32bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.36 registerWriteF64()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteF64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const double value,
    const short index )
```

Writes a double (64bit) register value.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>value</i> | The register value to write. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.37 registerWriteAscii()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteAscii (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const char * writeStr,
    const char writeEOL,
    const short index )
```

Writes a string register value.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeStr</i> | The zero terminated string to write. WriteStr will be limited to 239 characters and the terminating zero, totally 240 bytes. |
| <i>writeEOL</i> | <ul style="list-style-type: none"> • 0 Do NOT append End Of Line character (a null character) to the string. • 1 Append End Of Line character to the string. |
| <i>index</i> | Value index. Typically -1, but could be used to write a value in a mixed type register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write.

5.1.5.38 registerWriteRead()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteRead (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const void * writeData,
    const unsigned char writeSize,
    void * readData,
    unsigned char * readSize,
    const short index )
```

Writes and Reads a register value before returning.

Parameters

| | |
|------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeData</i> | Pointer to a data area from where the write value will be extracted. |
| <i>writeSize</i> | Size of write data area, ex. number of bytes to write. |
| <i>readData</i> | Pointer to a preallocated data area where the function will store the register read value. |
| <i>readSize</i> | Size of preallocated read data area, modified by the function to reflect the actual length of the read register value. The read register value may be truncated to fit into the allocated area. |
| <i>index</i> | Data index. Typically -1, but could be used to write/read data at/from a specific position in the register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

See also

[registerWriteReadU8](#), [registerWriteReadS8](#) etc.

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.39 registerWriteReadU8()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU8 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const unsigned char writeValue,
    unsigned char * readValue,
    const short index )
```

Writes and Reads an unsigned char (8bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to an unsigned char where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.40 registerWriteReadS8()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS8 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const signed char writeValue,
    signed char * readValue,
    const short index )
```

Writes and Reads a signed char (8bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to a signed char where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.41 registerWriteReadU16()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU16 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const unsigned short writeValue,
    unsigned short * readValue,
    const short index )
```

Writes and Reads an unsigned short (16bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to an unsigned short where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.42 registerWriteReadS16()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS16 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const signed short writeValue,
    signed short * readValue,
    const short index )
```

Writes and Reads a signed short (16bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to a signed short where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.43 registerWriteReadU32()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU32 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const unsigned long writeValue,
    unsigned long * readValue,
    const short index )
```

Writes and Reads an unsigned long (32bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to an unsigned long where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.44 registerWriteReadS32()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS32 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const signed long writeValue,
    signed long * readValue,
    const short index )
```

Writes and Reads a signed long (32bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to a signed long where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.45 registerWriteReadU64()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const unsigned long long writeValue,
    unsigned long long * readValue,
    const short index )
```

Writes and Reads an unsigned long long (64bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to an unsigned long long where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.46 registerWriteReadS64()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const signed long long writeValue,
    signed long long * readValue,
    const short index )
```

Writes and Reads a signed long long (64bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to a signed long long where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.47 registerWriteReadF32()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadF32 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const float writeValue,
    float * readValue,
    const short index )
```

Writes and Reads a float (32bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to a float where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.48 registerWriteReadF64()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadF64 (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const double writeValue,
    double * readValue,
    const short index )
```

Writes and Reads a double (64bit) register value.

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeValue</i> | The register value to write. |
| <i>readValue</i> | Pointer to a double where the function will store the register read value. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a value in a multi value register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.49 registerWriteReadAscii()

```
NKTPDLL_EXPORT RegisterResultTypes registerWriteReadAscii (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const char * writeStr,
    const char writeEOL,
    char * readStr,
    unsigned char * maxLen,
    const short index )
```

Writes and Reads a string register value.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>writeStr</i> | The zero terminated string to write. WriteStr will be limited to 239 characters and the terminating zero, totally 240 bytes. |
| <i>writeEOL</i> | <ul style="list-style-type: none"> • 0 Do NOT append End Of Line character (a null character) to the string. • 1 Append End Of Line character to the string. |
| <i>readStr</i> | Pointer to a preallocated string area where the function will store the register read value. |
| <i>maxLen</i> | Size of preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |
| <i>index</i> | Value index. Typically -1, but could be used to write and read a string in a mixed type register. Index is byte counted. Index ≥ 0 activates a read-modify-write sequence: The complete register content is being read, then the indexed content is modified and finally the complete content is written to the register. |

Returns

A status result value [tRegisterResultTypes](#)

Note

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated write followed by a dedicated read.

5.1.5.50 deviceGetType()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetType (
    const char * portname,
    const unsigned char devId,
    unsigned char * devType )
```

Returns the module type for a specific device id (module address).

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | Given device id to retrieve device type for (module type). |
| <i>devType</i> | Pointer to an unsigned char where the function stores the device type. |

Returns

A status result value [tDeviceResultTypes](#)

Note

Register address 0x61

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

Deprecated Should not be used in new applications, use [deviceGetTypeV2](#) instead.

5.1.5.51 deviceGetTypeV2()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetTypeV2 (
    const char * portname,
    const unsigned char devId,
    unsigned short * devType )
```

Returns the module type for a specific device id (module address).

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | Given device id to retrieve device type for (module type). |
| <i>devType</i> | Pointer to an unsigned short where the function stores the device type. |

Returns

A status result value [tDeviceResultTypes](#)

Note

Register address 0x61

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.52 deviceGetSysType()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetSysType (
    const char * portname,
    const unsigned char devId,
    unsigned char * sysType )
```

Returns the system type for a specific device id (module address).

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | Given device id to retrieve system type for (system type). |
| <i>sysType</i> | Pointer to an unsigned char where the function stores the system type. |

Returns

A status result value [tDeviceResultTypes](#)

Note

Register address 0x6B

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.53 deviceGetPartNumberStr()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetPartNumberStr (
    const char * portname,
    const unsigned char devId,
    char * partnumber,
    unsigned char * maxLen )
```

Returns the partnumber for a given device (module address).

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>partnumber</i> | Pointer to a preallocated string area where the function will store the partnumber. |
| <i>maxLen</i> | Size of preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |

Returns

A status result value [RegisterResultTypes](#)

Note

Register address 0x8E **Not all modules have a partnumber register.**

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

Deprecated No longer used - Should not be used in new applications.

5.1.5.54 deviceGetPCBVersion()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetPCBVersion (
    const char * portname,
    const unsigned char devId,
    unsigned char * PCBVersion )
```

Returns the PCB version for a given device (module address).

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>PCBVersion</i> | Pointer to a preallocated unsigned char where the function will store the PCB version. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x62

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.55 deviceGetStatusBits()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetStatusBits (
    const char * portname,
    const unsigned char devId,
    unsigned long * statusBits )
```

Returns the status bits for a given device (module address).

Parameters

| | |
|-------------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>statusBits</i> | Pointer to a preallocated unsigned long where the function will store the status bits |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x66

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.56 deviceGetErrorCode()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetErrorCode (
    const char * portname,
    const unsigned char devId,
    unsigned short * errorCode )
```

Returns the error code for a given device (module address).

Parameters

| | |
|------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>errorCode</i> | Pointer to a preallocated unsigned short where the function will store the error code. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x67

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.57 deviceGetBootloaderVersion()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetBootloaderVersion (
    const char * portname,
    const unsigned char devId,
    unsigned short * version )
```

Returns the bootloader version for a given device (module address).

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>version</i> | Pointer to a preallocated unsigned short where the function will store the bootloader version. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x6D

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.58 deviceGetBootloaderVersionStr()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetBootloaderVersionStr (
    const char * portname,
    const unsigned char devId,
    char * versionStr,
    unsigned char * maxLen )
```

Returns the bootloader version (string) for a given device (module address).

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>versionStr</i> | Pointer to a preallocated string area where the function will store the bootloader version. |
| <i>maxLen</i> | Size of preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x6D

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.59 deviceGetFirmwareVersion()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetFirmwareVersion (
    const char * portname,
    const unsigned char devId,
    unsigned short * version )
```

Returns the firmware version for a given device (module address).

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>version</i> | Pointer to a preallocated unsigned short where the function will store the firmware version. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x64

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.60 deviceGetFirmwareVersionStr()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetFirmwareVersionStr (
    const char * portname,
    const unsigned char devId,
    char * versionStr,
    unsigned char * maxLen )
```

Returns the firmware version (string) for a given device (module address).

Parameters

| | |
|-------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>versionStr</i> | Pointer to a preallocated string area where the function will store the firmware version. |
| <i>maxLen</i> | Size of preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x64

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.61 deviceGetModuleSerialNumberStr()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetModuleSerialNumberStr (
    const char * portname,
    const unsigned char devId,
    char * serialNumber,
    unsigned char * maxLen )
```

Returns the Module serialnumber (string) for a given device (module address).

Parameters

| | |
|---------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>serialNumber</i> | Pointer to a preallocated string area where the function will store the serialnumber version. |
| <i>maxLen</i> | Size of preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x65

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.62 **deviceGetPCBSerialNumberStr()**

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetPCBSerialNumberStr (
    const char * portname,
    const unsigned char devId,
    char * serialNumber,
    unsigned char * maxLen )
```

Returns the PCB serialnumber (string) for a given device (module address).

Parameters

| | |
|---------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>serialNumber</i> | Pointer to a preallocated string area where the function will store the serialnumber version. |
| <i>maxLen</i> | Size of preallocated string area, modified by the function to reflect the actual length of the returned string. The returned string may be truncated to fit into the allocated area. |

Returns

A status result value [tRegisterResultTypes](#)

Note

Register address 0x6E

It is not necessary to open the port, create the device or register before using this function, since it will do a dedicated read.

5.1.5.63 deviceCreate()

```
NKTPDLL_EXPORT DeviceResultTypes deviceCreate (
    const char * portname,
    const unsigned char devId,
    const char waitReady )
```

Creates a device in the internal devicelist. If the [openPorts](#) function has been called with the liveMode = 1 the kernel immediately starts to monitor the device.

Parameters

| | |
|------------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>waitReady</i> | <ul style="list-style-type: none">• 0 Don't wait for the device being ready.• 1 Wait up to 2 seconds for the device to complete its analyze cycle (All standard registers being successfully read). |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function.

5.1.5.64 deviceExists()

```
NKTPDLL_EXPORT DeviceResultTypes deviceExists (
    const char * portname,
    const unsigned char devId,
    unsigned char * exists )
```

Checks if a specific device already exists in the internal devicelist.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>exists</i> | <p>Pointer to an unsigned char where the function will store the exists status.</p> <ul style="list-style-type: none">• 0 Device does not exists.• 1 Device exists. |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function.

5.1.5.65 deviceRemove()

```
NKTPDLL_EXPORT DeviceResultTypes deviceRemove (  
    const char * portname,  
    const unsigned char devId )
```

Remove a specific device from the internal devicelist.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function.

5.1.5.66 deviceRemoveAll()

```
NKTPDLL_EXPORT DeviceResultTypes deviceRemoveAll (  
    const char * portname )
```

Remove all devices from the internal devicelist. No confirmation given, the list is simply cleared.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
|-----------------|--|

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function.

5.1.5.67 deviceGetAllTypes()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetAllTypes (  
    const char * portname,
```

```

    unsigned char * types,
    unsigned char * maxTypes )

```

Returns a list with device types (module types) from the internal devicelist.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>types</i> | Pointer to a preallocated area where the function stores the list of module types. The default list size is 256 bytes long (0-255) where each position indicates module address, containing 0 for no module or the module type for addresses having a module. ex. 00h 61h 62h 63h 64h 65h 00h 00h 00h 00h 00h 00h 00h 60h 00h 00h etc. Indicates module type 61h at address 1, module type 62h at address 2 etc. and module type 60h at address 15 |
| <i>maxTypes</i> | Pointer to an unsigned char giving the maximum number of types to retrieve. The returned list may be truncated to fit into the allocated area. |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function.

Deprecated Should not be used in new applications, use [deviceGetAllTypesV2](#) instead.

5.1.5.68 deviceGetAllTypesV2()

```

NKTPDLL_EXPORT DeviceResultTypes deviceGetAllTypesV2 (
    const char * portname,
    unsigned short * types,
    unsigned char * maxTypes )

```

Returns a list with device types (module types) from the internal devicelist.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>types</i> | Pointer to a preallocated area where the function stores the list of module types. The default list size is 512 bytes long (0-255 shorts) where each position indicates module address, containing 0 for no module or the module type for addresses having a module. ex. 0000h 0061h 0062h 0063h 0064h 0065h 0000h 0000h 0000h 0000h 0000h 0000h 0000h 0000h 0000h etc. Indicates module type 0061h at address 1, module type 0062h at address 2 etc. and module type 0060h at address 15 |
| <i>maxTypes</i> | Pointer to an unsigned char giving the maximum number of types to retrieve. The returned list may be truncated to fit into the allocated area. |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function.

5.1.5.69 deviceGetMode()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetMode (
    const char * portname,
    const unsigned char devId,
    unsigned char * devMode )
```

Returns the internal device mode for a specific device id (module address).

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | Given device id to retrieve device mode for. |
| <i>devMode</i> | Pointer to an DeviceModeTypes where the function stores the device mode value tDeviceModeTypes |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function and the device being already created, either automatically or with the [deviceCreate](#) function.

5.1.5.70 deviceGetLive()

```
NKTPDLL_EXPORT DeviceResultTypes deviceGetLive (
    const char * portname,
    const unsigned char devId,
    unsigned char * liveMode )
```

Returns the internal device live status for a specific device id (module address).

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | Given device id to retrieve liveMode. |
| <i>liveMode</i> | Pointer to an unsigned char where the function stores the live status. <ul style="list-style-type: none"> • 0 liveMode off • 1 liveMode on |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function and the device being already created, either automatically or with the [deviceCreate](#) function.

5.1.5.71 deviceSetLive()

```
NKTPDLL_EXPORT DeviceResultTypes deviceSetLive (
    const char * portname,
    const unsigned char devId,
    const unsigned char liveMode )
```

Sets the internal device live status for a specific device id (module address).

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | Given device id to set liveMode on. |
| <i>liveMode</i> | An unsigned char giving the new live status. <ul style="list-style-type: none">• 0 liveMode off• 1 liveMode on |

Returns

A status result value [tDeviceResultTypes](#)

Note

Requires the port being already opened with the [openPorts](#) function and the device being already created, either automatically or with the [deviceCreate](#) function.

5.1.5.72 registerCreate()

```
NKTPDLL_EXPORT RegisterResultTypes registerCreate (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    const RegisterPriorityTypes priority,
    const RegisterDataTypes dataType )
```

Creates a register in the internal registerlist. If the [openPorts](#) function has been called with the liveMode = 1 the kernel immediatedly starts to monitor the register.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>priority</i> | The tRegisterPriorityTypes (monitoring priority). |
| <i>dataType</i> | The tRegisterDataTypes , not used internally but could be used in a common callback function to determine data type. |

Returns

A status result value [tRegisterResultTypes](#)

5.1.5.73 registerExists()

```
NKTPDLL_EXPORT RegisterResultTypes registerExists (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId,
    unsigned char * exists )
```

Checks if a specific register already exists in the internal registerlist.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |
| <i>exists</i> | Pointer to an unsigned char where the function will store the exists status. <ul style="list-style-type: none"> • 0 Register does not exists. • 1 Register exists. |

Returns

A status result value [tRegisterResultTypes](#)

5.1.5.74 registerRemove()

```
NKTPDLL_EXPORT RegisterResultTypes registerRemove (
    const char * portname,
    const unsigned char devId,
    const unsigned char regId )
```

Remove a specific register from the internal registerlist.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regId</i> | The register id (register address). |

Returns

A status result value [tRegisterResultTypes](#)

5.1.5.75 registerRemoveAll()

```
NKTPDLL_EXPORT RegisterResultTypes registerRemoveAll (
    const char * portname,
    const unsigned char devId )
```

Remove all registers from the internal registerlist. No confirmation given, the list is simply cleared.

Parameters

| | |
|-----------------|--|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |

Returns

A status result value [tRegisterResultTypes](#)

5.1.5.76 registerGetAll()

```
NKTPDLL_EXPORT RegisterResultTypes registerGetAll (
    const char * portname,
    const unsigned char devId,
    unsigned char * regs,
    unsigned char * maxRegs )
```

Returns a list with register ids (register addresses) from the internal registerlist.

Parameters

| | |
|-----------------|---|
| <i>portname</i> | Zero terminated string giving the portname (case sensitive). |
| <i>devId</i> | The device id (module address). |
| <i>regs</i> | Pointer to a preallocated area where the function stores the list of register ids (register addresses). |
| <i>maxRegs</i> | Pointer to an unsigned char giving the maximum number of register ids to retrieve. Modified by the function to reflect the actual number of register ids returned. The returned list may be truncated to fit into the allocated area. |

Returns

A status result value [tRegisterResultTypes](#)

5.1.5.77 setCallbackPtrPortInfo()

```
NKTPDLL_EXPORT void setCallbackPtrPortInfo (
    PortStatusCallbackFuncPtr callback )
```

Enables/Disables callback for port status changes.

Parameters

| | |
|-----------------|---|
| <i>callback</i> | The PortStatusCallbackFuncPtr function pointer. Disable callbacks by parsing in a zero value. |
|-----------------|---|

5.1.5.78 setCallbackPtrDeviceInfo()

```
NKTPDLL_EXPORT void setCallbackPtrDeviceInfo (
    DeviceStatusCallbackFuncPtr callback )
```

Enables/Disables callback for device status changes.

Parameters

| | |
|-----------------|---|
| <i>callback</i> | The DeviceStatusCallbackFuncPtr function pointer. Disable callbacks by parsing in a zero value. |
|-----------------|---|

5.1.5.79 setCallbackPtrRegisterInfo()

```
NKTPDLL_EXPORT void setCallbackPtrRegisterInfo (
    RegisterStatusCallbackFuncPtr callback )
```

Enables/Disables callback for register status changes.

Parameters

| | |
|-----------------|---|
| <i>callback</i> | The RegisterStatusCallbackFuncPtr function pointer. Disable callbacks by parsing in a zero value. |
|-----------------|---|

5.1.5.80 setLVUserEventPortInfo()

```
NKTPDLL_EXPORT void setLVUserEventPortInfo (
    unsigned long * lvUserEventRef )
```

Enables/Disables labView user events for port status changes. Disable events by parsing in a zero value.

Parameters

| | |
|-----------------------|--|
| <i>lvUserEventRef</i> | A LabView "MagicCookie" to identify useevent type. |
|-----------------------|--|

5.1.5.81 setLVUserEventDeviceInfo()

```
NKTPDLL_EXPORT void setLVUserEventDeviceInfo (
    unsigned long * lvUserEventRef )
```

Enables/Disables labView user events for device status changes. Disable events by parsing in a zero value.

Parameters

| | |
|-----------------------|---|
| <i>lvUserEventRef</i> | A LabView "MagicCookie" to identify userevent type. |
|-----------------------|---|

5.1.5.82 setLVUserEventRegisterInfo()

```
NKTPDLL_EXPORT void setLVUserEventRegisterInfo (
    unsigned long * lvUserEventRef )
```

Enables/Disables labView user events for register status changes. Disable events by parsing in a zero value.

Parameters

| | |
|-----------------------|---|
| <i>lvUserEventRef</i> | A LabView "MagicCookie" to identify userevent type. |
|-----------------------|---|

5.2 NKTPDLL.h

[Go to the documentation of this file.](#)

```
00001 #ifndef NKTPDLL_H
00002 #define NKTPDLL_H
00003
00013 #if defined(NKTPDLL_LIBRARY)
00014     #ifdef __cplusplus
00015         #define NKTPDLL_EXPORT extern "C" __declspec(dllexport)
00016     #else
00017         #define NKTPDLL_EXPORT __declspec(dllexport)
00018     #endif
00019 #else
00020     #ifdef __cplusplus
00021         #define NKTPDLL_EXPORT extern "C" __declspec(dllimport)
00022     #else
00023         #define NKTPDLL_EXPORT __declspec(dllimport)
00024     #endif
00025 #endif
00026
00027 #ifdef __cplusplus
00028 namespace NKTPDLL
00029 {
00030 #endif
00031
00035 enum tPortResultTypes
00036 {
00037     OPSuccess = 0,
00038     OPFailed = 1,
00039     OPPortNotFound = 2,
00040     OPNoDevices = 3,
00041     OPApplicationBusy = 4
00042 };
00043 typedef unsigned char PortResultTypes;
00044
00048 enum tP2PPortResultTypes
00049 {
00050     P2PSuccess = 0,
00051     P2PInvalidPortname = 1,
00052     P2PInvalidLocalIP = 2,
00053     P2PInvalidRemoteIP = 3,
```

```
00054     P2PPortnameNotFound = 4,
00055     P2PPortnameExists = 5,
00056     P2PApplicationBusy = 6
00057 };
00058 typedef unsigned char P2PPortResultTypes;
00059
00063 enum tDeviceResultTypes
00064 {
00065     DevResultSuccess = 0,
00066     DevResultWaitTimeout = 1,
00067     DevResultFailed = 2,
00068     DevResultDeviceNotFound = 3,
00069     DevResultPortNotFound = 4,
00070     DevResultPortOpenError = 5,
00071     DevResultApplicationBusy = 6
00072 };
00073 typedef unsigned char DeviceResultTypes;
00074
00078 enum tDeviceModeTypes
00079 {
00080     DevModeDisabled = 0,
00081     DevModeAnalyzeInit = 1,
00082     DevModeAnalyze = 2,
00083     DevModeNormal = 3,
00084     DevModeLogDownload = 4,
00085     DevModeError = 5,
00086     DevModeTimeout = 6,
00087     DevModeUpload = 7,
00088 };
00089 typedef unsigned char DeviceModeTypes;
00090
00094 enum tRegisterResultTypes
00095 {
00096     RegResultSuccess = 0,
00097     RegResultReadError = 1,
00098     RegResultFailed = 2,
00099     RegResultBusy = 3,
00100     RegResultNacked = 4,
00101     RegResultCRCErr = 5,
00102     RegResultTimeout = 6,
00103     RegResultComError = 7,
00104     RegResultTypeError = 8,
00105     RegResultIndexError = 9,
00106     RegResultPortClosed = 10,
00107     RegResultRegisterNotFound = 11,
00108     RegResultDeviceNotFound = 12,
00109     RegResultPortNotFound = 13,
00110     RegResultPortOpenError = 14,
00111     RegResultApplicationBusy = 15
00112 };
00113 typedef unsigned char RegisterResultTypes;
00114
00118 enum tRegisterDataTypes
00119 {
00120     RegData_Unknown = 0,
00121     RegData_Mixed = 1,
00122     RegData_U8 = 2,
00123     RegData_S8 = 3,
00124     RegData_U16 = 4,
00125     RegData_S16 = 5,
00126     RegData_U32 = 6,
00127     RegData_S32 = 7,
00128     RegData_F32 = 8,
00129     RegData_U64 = 9,
00130     RegData_S64 = 10,
00131     RegData_F64 = 11,
00132     RegData_Ascii = 12,
00133     RegData_Paramset = 13,
00134     RegData_B8 = 14,
00135     RegData_H8 = 15,
00136     RegData_B16 = 16,
00137     RegData_H16 = 17,
00138     RegData_B32 = 18,
00139     RegData_H32 = 19,
00140     RegData_B64 = 20,
00141     RegData_H64 = 21,
00142     RegData_DateTime = 22,
00143 };
00144 typedef unsigned char RegisterDataTypes;
00145
00149 enum tRegisterPriorityTypes
00150 {
00151     RegPriority_Low = 0,
00152     RegPriority_High = 1
00153 };
00154 typedef unsigned char RegisterPriorityTypes;
00155
```

```

00159 enum tPortStatusTypes
00160 {
00161     PortStatusUnknown = 0,
00162     PortOpening = 1,
00163     PortOpened = 2,
00164     PortOpenFail = 3,
00165     PortScanStarted = 4,
00166     PortScanProgress = 5,
00167     PortScanDeviceFound = 6,
00168     PortScanEnded = 7,
00169     PortClosing = 8,
00170     PortClosed = 9,
00171     PortReady = 10
00172 };
00173 typedef unsigned char PortStatusTypes;
00174
00175 enum tDeviceStatusTypes
00176 {
00177     DeviceModeChanged = 0,
00178     DeviceLiveChanged = 1,
00179     DeviceTypeChanged = 2,
00180     DevicePartNumberChanged = 3,
00181     DevicePCBVersionChanged = 4,
00182     DeviceStatusBitsChanged = 5,
00183     DeviceErrorCodeChanged = 6,
00184     DeviceBlVerChanged = 7,
00185     DeviceFwVerChanged = 8,
00186     DeviceModuleSerialChanged = 9,
00187     DevicePCBSerialChanged = 10,
00188     DeviceSysTypeChanged = 11
00189 };
00190 typedef unsigned char DeviceStatusTypes;
00191
00192 enum tRegisterStatusTypes
00193 {
00194     RegSuccess = 0,
00195     RegBusy = 1,
00196     RegNacked = 2,
00197     RegCRCError = 3,
00198     RegTimeout = 4,
00199     RegComError = 5
00200 };
00201 typedef unsigned char RegisterStatusTypes;
00202
00203 #pragma pack(1)
00204 typedef struct tDateTimeStruct
00205 {
00206     unsigned char Sec;
00207     unsigned char Min;
00208     unsigned char Hour;
00209     unsigned char Day;
00210     unsigned char Month;
00211     unsigned char Year;
00212 } DateTimeType;
00213 #pragma pack()
00214
00215 enum tParamSetUnitTypes
00216 {
00217     // Unit codes
00218     UnitNone = 0,
00219     UnitmV = 1,
00220     UnitV = 2,
00221     UnituA = 3,
00222     UnitmA = 4,
00223     UnitA = 5,
00224     UnituW = 6,
00225     UnitcmW = 7,
00226     UnitdmW = 8,
00227     UnitmW = 9,
00228     UnitW = 10,
00229     UnitmC = 11,
00230     UnitcC = 12,
00231     UnitdC = 13,
00232     Unitpm = 14,
00233     Unitdnm = 15,
00234     Unitnm = 16,
00235     UnitPerCent = 17,
00236     UnitPerMille = 18,
00237     UnitcmA = 19,
00238     UnitdmA = 20,
00239     UnitRPM = 21,
00240     UnitdBm = 22,
00241     UnitcBm = 23,
00242     UnitmBm = 24,
00243     UnitdB = 25,
00244     UnitcB = 26,
00245     UnitmB = 27,

```

```

00258     Unitdpm = 28,
00259     UnitcV = 29,
00260     UnitdV = 30,
00261     Unitlm = 31,
00262     Unitdlm = 32,
00263     Unitclm = 33,
00264     Unitmlm = 34,
00265     UnitHz = 35,
00266     UnitkHz = 36,
00267     UnitMHz = 37,
00268     UnitSec = 38,
00269     UnitmSec = 39,
00270     UnituSec = 40,
00271     UnitdA = 41,
00272     UnitcA = 42,
00273     UnitduA = 43,
00274     UnitcuA = 44,
00275     UnitnA = 45,
00276     UnitdW = 46,
00277     UnitcW = 47,
00278     UnitpA = 48
00279 };
00280 typedef unsigned char ParamSetUnitTypes;
00281
00282 #pragma pack(1)
00283 typedef struct tParamSetStruct
00284 {
00285     ParamSetUnitTypes Unit;
00286     unsigned char ErrorHandler;
00287     unsigned short StartVal;
00288     unsigned short FactoryVal;
00289     unsigned short ULimit;
00290     unsigned short LLimit;
00291     signed short Numerator;
00292     signed short Denominator;
00293     signed short Offset;
00294 } ParameterSetType;
00295 #pragma pack()
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305 /*****
00306  * Port functions
00307  *****/
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421

```

```

00422
00431 NKTPDLL_EXPORT void setSpecificBusScanningRange(const char specificScanning, const unsigned char
startAddress, const unsigned char endAddress);
00432 typedef void (__cdecl *SetSpecificBusScanningRangeFuncPtr)(const char specificScanning, const unsigned
char startAddress, const unsigned char endAddress);
00433
00434
00442 NKTPDLL_EXPORT void getSpecificBusScanningRange(char *specificScanning, unsigned char *startAddress,
unsigned char *endAddress);
00443 typedef unsigned char (__cdecl *GetSpecificBusScanningRangeFuncPtr)(char *specificScanning, unsigned
char *startAddress, unsigned char *endAddress);
00444
00445
00452 NKTPDLL_EXPORT PortResultTypes getPortStatus(const char *portname, PortStatusTypes *portStatus);
00453 typedef PortResultTypes (__cdecl *GetPortStatusFuncPtr)(const char *portname, PortStatusTypes
*portStatus);
00454
00455
00463 NKTPDLL_EXPORT PortResultTypes getPortErrorMsg(const char *portname, char *errorMessage, unsigned
short *maxLen);
00464 typedef PortResultTypes (__cdecl *GetPortErrorMsgFuncPtr)(const char *portname, char *errorMessage,
unsigned short *maxLen);
00465
00466
00469
00470 /*****
* Dedicated - Register read functions
00471
*****/
00490 NKTPDLL_EXPORT RegisterResultTypes registerRead(const char *portname, const unsigned char devId, const
unsigned char regId, void *readData, unsigned char *readSize, const short index);
00491 typedef RegisterResultTypes (__cdecl *RegisterReadFuncPtr)(const char *portname, const unsigned char
devId, const unsigned char regId, void *readData, unsigned char *readSize, const short index);
00492
00503 NKTPDLL_EXPORT RegisterResultTypes registerReadU8(const char *portname, const unsigned char devId,
const unsigned char regId, unsigned char *value, const short index);
00504 typedef RegisterResultTypes (__cdecl *RegisterReadU8FuncPtr)(const char *portname, const unsigned char
devId, const unsigned char regId, unsigned char *value, const short index);
00505
00516 NKTPDLL_EXPORT RegisterResultTypes registerReadS8(const char *portname, const unsigned char devId,
const unsigned char regId, signed char *value, const short index);
00517 typedef RegisterResultTypes (__cdecl *RegisterReadS8FuncPtr)(const char *portname, const unsigned char
devId, const unsigned char regId, signed char *value, const short index);
00518
00529 NKTPDLL_EXPORT RegisterResultTypes registerReadU16(const char *portname, const unsigned char devId,
const unsigned char regId, unsigned short *value, const short index);
00530 typedef RegisterResultTypes (__cdecl *RegisterReadU16FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, unsigned short *value, const short index);
00531
00542 NKTPDLL_EXPORT RegisterResultTypes registerReadS16(const char *portname, const unsigned char devId,
const unsigned char regId, signed short *value, const short index);
00543 typedef RegisterResultTypes (__cdecl *RegisterReadS16FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, signed short *value, const short index);
00544
00555 NKTPDLL_EXPORT RegisterResultTypes registerReadU32(const char *portname, const unsigned char devId,
const unsigned char regId, unsigned long *value, const short index);
00556 typedef RegisterResultTypes (__cdecl *RegisterReadU32FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, unsigned long *value, const short index);
00557
00568 NKTPDLL_EXPORT RegisterResultTypes registerReadS32(const char *portname, const unsigned char devId,
const unsigned char regId, signed long *value, const short index);
00569 typedef RegisterResultTypes (__cdecl *RegisterReadS32FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, signed long *value, const short index);
00570
00581 NKTPDLL_EXPORT RegisterResultTypes registerReadU64(const char *portname, const unsigned char devId,
const unsigned char regId, unsigned long long *value, const short index);
00582 typedef RegisterResultTypes (__cdecl *RegisterReadU64FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, unsigned long long *value, const short index);
00583
00594 NKTPDLL_EXPORT RegisterResultTypes registerReadS64(const char *portname, const unsigned char devId,
const unsigned char regId, signed long long *value, const short index);
00595 typedef RegisterResultTypes (__cdecl *RegisterReadS64FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, signed long long *value, const short index);
00596
00607 NKTPDLL_EXPORT RegisterResultTypes registerReadF32(const char *portname, const unsigned char devId,
const unsigned char regId, float *value, const short index);
00608 typedef RegisterResultTypes (__cdecl *RegisterReadF32FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, float *value, const short index);
00609
00620 NKTPDLL_EXPORT RegisterResultTypes registerReadF64(const char *portname, const unsigned char devId,
const unsigned char regId, double *value, const short index);
00621 typedef RegisterResultTypes (__cdecl *RegisterReadF64FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, double *value, const short index);
00622
00634 NKTPDLL_EXPORT RegisterResultTypes registerReadAscii(const char *portname, const unsigned char devId,
const unsigned char regId, char *readStr, unsigned char *maxLen, const short index);
00635 typedef RegisterResultTypes (__cdecl *RegisterReadAsciiFuncPtr)(const char *portname, const unsigned

```

```

char devId, const unsigned char regId, char *readStr, unsigned char *maxLen, const short index);
00636
00639
/*****
00640 * Dedicated - Register write functions
00641 *****/
00662 NKTPDLL_EXPORT RegisterResultTypes registerWrite(const char *portname, const unsigned char devId,
const unsigned char regId, const void *writeData, const unsigned char writeSize, const short index);
00663 typedef RegisterResultTypes (__cdecl *RegisterWriteFuncPtr)(const char *portname, const unsigned char
devId, const unsigned char regId, const void *writeData, const unsigned char writeSize, const short
index);
00664
00677 NKTPDLL_EXPORT RegisterResultTypes registerWriteU8(const char *portname, const unsigned char devId,
const unsigned char regId, const unsigned char value, const short index);
00678 typedef RegisterResultTypes (__cdecl *RegisterWriteU8FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const unsigned char value, const short index);
00679
00692 NKTPDLL_EXPORT RegisterResultTypes registerWriteS8(const char *portname, const unsigned char devId,
const unsigned char regId, const signed char value, const short index);
00693 typedef RegisterResultTypes (__cdecl *RegisterWriteS8FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const signed char value, const short index);
00694
00707 NKTPDLL_EXPORT RegisterResultTypes registerWriteU16(const char *portname, const unsigned char devId,
const unsigned char regId, const unsigned short value, const short index);
00708 typedef RegisterResultTypes (__cdecl *RegisterWriteU16FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const unsigned short value, const short index);
00709
00722 NKTPDLL_EXPORT RegisterResultTypes registerWriteS16(const char *portname, const unsigned char devId,
const unsigned char regId, const signed short value, const short index);
00723 typedef RegisterResultTypes (__cdecl *RegisterWriteS16FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const signed short value, const short index);
00724
00737 NKTPDLL_EXPORT RegisterResultTypes registerWriteU32(const char *portname, const unsigned char devId,
const unsigned char regId, const unsigned long value, const short index);
00738 typedef RegisterResultTypes (__cdecl *RegisterWriteU32FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const unsigned long value, const short index);
00739
00752 NKTPDLL_EXPORT RegisterResultTypes registerWriteS32(const char *portname, const unsigned char devId,
const unsigned char regId, const signed long value, const short index);
00753 typedef RegisterResultTypes (__cdecl *RegisterWriteS32FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const signed long value, const short index);
00754
00767 NKTPDLL_EXPORT RegisterResultTypes registerWriteU64(const char *portname, const unsigned char devId,
const unsigned char regId, const unsigned long long value, const short index);
00768 typedef RegisterResultTypes (__cdecl *RegisterWriteU64FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const unsigned long long value, const short index);
00769
00782 NKTPDLL_EXPORT RegisterResultTypes registerWriteS64(const char *portname, const unsigned char devId,
const unsigned char regId, const signed long long value, const short index);
00783 typedef RegisterResultTypes (__cdecl *RegisterWriteS64FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const signed long long value, const short index);
00784
00797 NKTPDLL_EXPORT RegisterResultTypes registerWriteF32(const char *portname, const unsigned char devId,
const unsigned char regId, const float value, const short index);
00798 typedef RegisterResultTypes (__cdecl *RegisterWriteF32FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const float value, const short index);
00799
00812 NKTPDLL_EXPORT RegisterResultTypes registerWriteF64(const char *portname, const unsigned char devId,
const unsigned char regId, const double value, const short index);
00813 typedef RegisterResultTypes (__cdecl *RegisterWriteF64FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const double value, const short index);
00814
00829 NKTPDLL_EXPORT RegisterResultTypes registerWriteAscii(const char *portname, const unsigned char devId,
const unsigned char regId, const char *writeStr, const char writeEOL, const short index);
00830 typedef RegisterResultTypes (__cdecl *RegisterWriteAsciiFuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const char *writeStr, const char writeEOL, const short index);
00831
00834
/*****
00835 * Dedicated - Register write/read functions (A write immediately followed by a read)
00836 *****/
00859 NKTPDLL_EXPORT RegisterResultTypes registerWriteRead(const char *portname, const unsigned char devId,
const unsigned char regId, const void *writeData, const unsigned char writeSize, void *readData,
unsigned char *readSize, const short index);
00860 typedef RegisterResultTypes (__cdecl *RegisterWriteReadFuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const void *writeData, const unsigned char writeSize, void
*readData, unsigned char *readSize, const short index);
00861
00875 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU8(const char *portname, const unsigned char
devId, const unsigned char regId, const unsigned char writeValue, unsigned char *readValue, const
short index);
00876 typedef RegisterResultTypes (__cdecl *RegisterWriteReadU8FuncPtr)(const char *portname, const unsigned
char devId, const unsigned char regId, const unsigned char writeValue, unsigned char *readValue, const
short index);
00877

```



```

00891 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS8(const char *portname, const unsigned char
    devId, const unsigned char regId, const signed char writeValue, signed char *readValue, const short
    index);
00892 typedef RegisterResultTypes (__cdecl *RegisterWriteReadS8FuncPtr)(const char *portname, const unsigned
    char devId, const unsigned char regId, const signed char writeValue, signed char *readValue, const
    short index);
00893
00907 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU16(const char *portname, const unsigned char
    devId, const unsigned char regId, const unsigned short writeValue, unsigned short *readValue, const
    short index);
00908 typedef RegisterResultTypes (__cdecl *RegisterWriteReadU16FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const unsigned short writeValue, unsigned short
    *readValue, const short index);
00909
00923 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS16(const char *portname, const unsigned char
    devId, const unsigned char regId, const signed short writeValue, signed short *readValue, const short
    index);
00924 typedef RegisterResultTypes (__cdecl *RegisterWriteReadS16FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const signed short writeValue, signed short
    *readValue, const short index);
00925
00939 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU32(const char *portname, const unsigned char
    devId, const unsigned char regId, const unsigned long writeValue, unsigned long *readValue, const
    short index);
00940 typedef RegisterResultTypes (__cdecl *RegisterWriteReadU32FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const unsigned long writeValue, unsigned long
    *readValue, const short index);
00941
00955 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS32(const char *portname, const unsigned char
    devId, const unsigned char regId, const signed long writeValue, signed long *readValue, const short
    index);
00956 typedef RegisterResultTypes (__cdecl *RegisterWriteReadS32FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const signed long writeValue, signed long *readValue,
    const short index);
00957
00971 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadU64(const char *portname, const unsigned char
    devId, const unsigned char regId, const unsigned long long writeValue, unsigned long long *readValue,
    const short index);
00972 typedef RegisterResultTypes (__cdecl *RegisterWriteReadU64FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const unsigned long long writeValue, unsigned long
    long *readValue, const short index);
00973
00987 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadS64(const char *portname, const unsigned char
    devId, const unsigned char regId, const signed long long writeValue, signed long long *readValue,
    const short index);
00988 typedef RegisterResultTypes (__cdecl *RegisterWriteReadS64FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const signed long long writeValue, signed long long
    *readValue, const short index);
00989
01003 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadF32(const char *portname, const unsigned char
    devId, const unsigned char regId, const float writeValue, float *readValue, const short index);
01004 typedef RegisterResultTypes (__cdecl *RegisterWriteReadF32FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const float writeValue, float *readValue, const short
    index);
01005
01019 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadF64(const char *portname, const unsigned char
    devId, const unsigned char regId, const double writeValue, double *readValue, const short index);
01020 typedef RegisterResultTypes (__cdecl *RegisterWriteReadF64FuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const double writeValue, double *readValue, const
    short index);
01021
01038 NKTPDLL_EXPORT RegisterResultTypes registerWriteReadAscii(const char *portname, const unsigned char
    devId, const unsigned char regId, const char* writeStr, const char writeEOL, char *readStr, unsigned
    char *maxLen, const short index);
01039 typedef RegisterResultTypes (__cdecl *RegisterWriteReadAsciiFuncPtr)(const char *portname, const
    unsigned char devId, const unsigned char regId, const char* writeStr, const char writeEOL, char
    *readStr, unsigned char *maxLen, const short index);
01040
01041
01044
01045 /*****
    * Dedicated - Device functions
01046 *****/
01064 NKTPDLL_EXPORT DeviceResultTypes deviceGetType(const char *portname, const unsigned char devId,
    unsigned char *devType);
01065 typedef DeviceResultTypes (__cdecl *DeviceGetTypeFuncPtr)(const char *portname, const unsigned char
    devId, unsigned char *devType);
01066
01076 NKTPDLL_EXPORT DeviceResultTypes deviceGetTypeV2(const char *portname, const unsigned char devId,
    unsigned short *devType);
01077 typedef DeviceResultTypes (__cdecl *DeviceGetTypeV2FuncPtr)(const char *portname, const unsigned char
    devId, unsigned short *devType);
01078
01088 NKTPDLL_EXPORT DeviceResultTypes deviceGetSysType(const char *portname, const unsigned char devId,
    unsigned char *sysType);
01089 typedef DeviceResultTypes (__cdecl *DeviceGetSysTypeFuncPtr)(const char *portname, const unsigned char

```

```

    devId, unsigned char *sysType);
01090
01102 NKTPDLL_EXPORT DeviceResultTypes deviceGetPartNumberStr(const char *portname, const unsigned char
    devId, char *partnumber, unsigned char *maxLen);
01103 typedef DeviceResultTypes (__cdecl *DeviceGetPartNumberStrFuncPtr)(const char *portname, const
    unsigned char devId, char *partnumber, unsigned char *maxLen);
01104
01114 NKTPDLL_EXPORT DeviceResultTypes deviceGetPCBVersion(const char *portname, const unsigned char devId,
    unsigned char *PCBVersion);
01115 typedef DeviceResultTypes (__cdecl *DeviceGetPCBVersionFuncPtr)(const char *portname, const unsigned
    char devId, unsigned char *PCBVersion);
01116
01126 NKTPDLL_EXPORT DeviceResultTypes deviceGetStatusBits(const char *portname, const unsigned char devId,
    unsigned long *statusBits);
01127 typedef DeviceResultTypes (__cdecl *DeviceGetStatusBitsFuncPtr)(const char *portname, const unsigned
    char devId, unsigned long *statusBits);
01128
01138 NKTPDLL_EXPORT DeviceResultTypes deviceGetErrorCode(const char *portname, const unsigned char devId,
    unsigned short *errorCode);
01139 typedef DeviceResultTypes (__cdecl *DeviceGetErrorCodeFuncPtr)(const char *portname, const unsigned
    char devId, unsigned short *errorCode);
01140
01150 NKTPDLL_EXPORT DeviceResultTypes deviceGetBootloaderVersion(const char *portname, const unsigned char
    devId, unsigned short *version);
01151 typedef DeviceResultTypes (__cdecl *DeviceGetBootloaderVersionFuncPtr)(const char *portname, const
    unsigned char devId, unsigned short *version);
01152
01163 NKTPDLL_EXPORT DeviceResultTypes deviceGetBootloaderVersionStr(const char *portname, const unsigned
    char devId, char *versionStr, unsigned char *maxLen);
01164 typedef DeviceResultTypes (__cdecl *DeviceGetBootloaderVersionStrFuncPtr)(const char *portname, const
    unsigned char devId, char *versionStr, unsigned char *maxLen);
01165
01175 NKTPDLL_EXPORT DeviceResultTypes deviceGetFirmwareVersion(const char *portname, const unsigned char
    devId, unsigned short *version);
01176 typedef DeviceResultTypes (__cdecl *DeviceGetFirmwareVersionFuncPtr)(const char *portname, const
    unsigned char devId, unsigned short *version);
01177
01188 NKTPDLL_EXPORT DeviceResultTypes deviceGetFirmwareVersionStr(const char *portname, const unsigned char
    devId, char *versionStr, unsigned char *maxLen);
01189 typedef DeviceResultTypes (__cdecl *DeviceGetFirmwareVersionStrFuncPtr)(const char *portname, const
    unsigned char devId, char *versionStr, unsigned char *maxLen);
01190
01201 NKTPDLL_EXPORT DeviceResultTypes deviceGetModuleSerialNumberStr(const char *portname, const unsigned
    char devId, char *serialNumber, unsigned char *maxLen);
01202 typedef DeviceResultTypes (__cdecl *DeviceGetModuleSerialNumberStrFuncPtr)(const char *portname, const
    unsigned char devId, char *serialNumber, unsigned char *maxLen);
01203
01214 NKTPDLL_EXPORT DeviceResultTypes deviceGetPCBSerialNumberStr(const char *portname, const unsigned char
    devId, char *serialNumber, unsigned char *maxLen);
01215 typedef DeviceResultTypes (__cdecl *DeviceGetPCBSerialNumberStrFuncPtr)(const char *portname, const
    unsigned char devId, char *serialNumber, unsigned char *maxLen);
01216
01217
01220
    /*****
01221  * Callback - Device functions
01222
    *****/
01238 NKTPDLL_EXPORT DeviceResultTypes deviceCreate(const char *portname, const unsigned char devId, const
    char waitReady);
01239 typedef DeviceResultTypes (__cdecl *DeviceCreateFuncPtr)(const char *portname, const unsigned char
    devId, const char waitReady);
01240
01251 NKTPDLL_EXPORT DeviceResultTypes deviceExists(const char *portname, const unsigned char devId,
    unsigned char *exists);
01252 typedef DeviceResultTypes (__cdecl *DeviceExistsFuncPtr)(const char *portname, const unsigned char
    devId, unsigned char *exists);
01253
01261 NKTPDLL_EXPORT DeviceResultTypes deviceRemove(const char *portname, const unsigned char devId);
01262 typedef DeviceResultTypes (__cdecl *DeviceRemoveFuncPtr)(const char *portname, const unsigned char
    devId);
01263
01270 NKTPDLL_EXPORT DeviceResultTypes deviceRemoveAll(const char *portname);
01271 typedef DeviceResultTypes (__cdecl *DeviceRemoveAllFuncPtr)(const char *portname);
01272
01285 NKTPDLL_EXPORT DeviceResultTypes deviceGetAllTypes(const char *portname, unsigned char *types,
    unsigned char *maxTypes);
01286 typedef DeviceResultTypes (__cdecl *DeviceGetAllTypesFuncPtr)(const char *portname, unsigned char
    *types, unsigned char *maxTypes);
01287
01299 NKTPDLL_EXPORT DeviceResultTypes deviceGetAllTypesV2(const char *portname, unsigned short *types,
    unsigned char *maxTypes);
01300 typedef DeviceResultTypes (__cdecl *DeviceGetAllTypesV2FuncPtr)(const char *portname, unsigned short
    *types, unsigned char *maxTypes);
01301
01310 NKTPDLL_EXPORT DeviceResultTypes deviceGetMode(const char *portname, const unsigned char devId,
    unsigned char *devMode);

```

```

01311 typedef DeviceResultTypes (__cdecl *DeviceGetModeFuncPtr)(const char *portname, const unsigned char
devId, unsigned char *devMode);
01312
01323 NKTPDLL_EXPORT DeviceResultTypes deviceGetLive(const char *portname, const unsigned char devId,
unsigned char *liveMode);
01324 typedef DeviceResultTypes (__cdecl *DeviceGetLiveFuncPtr)(const char *portname, const unsigned char
devId, unsigned char *liveMode);
01325
01336 NKTPDLL_EXPORT DeviceResultTypes deviceSetLive(const char *portname, const unsigned char devId, const
unsigned char liveMode);
01337 typedef DeviceResultTypes (__cdecl *DeviceSetLiveFuncPtr)(const char *portname, const unsigned char
devId, const unsigned char liveMode);
01338
01339
01342
/*****
01343 * Callback - Register functions
01344 *****/
01359 NKTPDLL_EXPORT RegisterResultTypes registerCreate(const char *portname, const unsigned char devId,
const unsigned char regId, const RegisterPriorityTypes priority, const RegisterDataTypes dataType);
01360 typedef RegisterResultTypes (__cdecl *RegisterCreateFuncPtr)(const char *portname, const unsigned char
devId, const unsigned char regId, const RegisterPriorityTypes priority, const RegisterDataTypes
dataType);
01361
01372 NKTPDLL_EXPORT RegisterResultTypes registerExists(const char *portname, const unsigned char devId,
const unsigned char regId, unsigned char *exists);
01373 typedef RegisterResultTypes (__cdecl *RegisterExistsFuncPtr)(const char *portname, const unsigned char
devId, const unsigned char regId, unsigned char *exists);
01374
01382 NKTPDLL_EXPORT RegisterResultTypes registerRemove(const char *portname, const unsigned char devId,
const unsigned char regId);
01383 typedef RegisterResultTypes (__cdecl *RegisterRemoveFuncPtr)(const char *portname, const unsigned char
devId, const unsigned char regId);
01384
01391 NKTPDLL_EXPORT RegisterResultTypes registerRemoveAll(const char *portname, const unsigned char devId);
01392 typedef RegisterResultTypes (__cdecl *RegisterRemoveAllFuncPtr)(const char *portname, const unsigned
char devId);
01393
01403 NKTPDLL_EXPORT RegisterResultTypes registerGetAll(const char *portname, const unsigned char devId,
unsigned char *regs, unsigned char *maxRegs);
01404 typedef RegisterResultTypes (__cdecl *RegisterGetAllFuncPtr)(const char *portname, const unsigned char
devId, unsigned char *regs, unsigned char *maxRegs);
01405
01406
01409
/*****
01410 * Callback - Support functions
01411 *****/
01426 typedef void (__cdecl *PortStatusCallbackFuncPtr)(const char* portname,           // current port
name
01427                                     const PortStatusTypes status,           // current port
status
01428                                     const unsigned char curScanAdr,           // current scanned
address or device found address
01429                                     const unsigned char maxScanAdr,           // total addresses
to scan
01430                                     const unsigned short foundType);           // device found
type
01431
01436 NKTPDLL_EXPORT void setCallbackPtrPortInfo(PortStatusCallbackFuncPtr callback);
01437 typedef void (__cdecl *SetCallbackPtrPortInfoFuncPtr)(PortStatusCallbackFuncPtr callback);
01438
01439
01450 typedef void (__cdecl *DeviceStatusCallbackFuncPtr)(const char* portname,           //
current port name
01451                                     const unsigned char devId,           //
current device id
01452                                     const DeviceStatusTypes status,           //
current device status
01453                                     const unsigned char devDataLen,           //
number of bytes in devData
01454                                     const void* devData);           //
device data as specified in status
01455
01460 NKTPDLL_EXPORT void setCallbackPtrDeviceInfo(DeviceStatusCallbackFuncPtr callback);
01461 typedef void (__cdecl *SetCallbackPtrDeviceInfoFuncPtr)(DeviceStatusCallbackFuncPtr callback);
01462
01463
01464
01477 typedef void (__cdecl *RegisterStatusCallbackFuncPtr)(const char* portname,           //
current port name
01478                                     const unsigned char devId,           //
current device id
01479                                     const unsigned char regId,           //
current device id

```

```

01480                                     const RegisterStatusTypes status,           //
current register status
01481                                     const RegisterDataTypes regType,           //
current register type
01482                                     const unsigned char regDataLen,           //
number of bytes in regData
01483                                     const void *regData);                       //
register data
01484
01489 NKTPDLL_EXPORT void setCallbackPtrRegisterInfo(RegisterStatusCallbackFuncPtr callback);
01490 typedef void (__cdecl *SetCallbackPtrRegisterInfoFuncPtr)(RegisterStatusCallbackFuncPtr callback);
01491
01494
01495 /*****
* LabView - Support functions
01496
*****
01504 #pragma pack(1)
01505 typedef struct lvPortStatusStruct
01506 {
01507     char portname[32];
01508     PortStatusTypes status;
01509     unsigned char curScanAdr;
01510     unsigned char maxScanAdr;
01511     unsigned short foundType;
01512 } LabViewPortStatusType;
01513 #pragma pack()
01514
01519 NKTPDLL_EXPORT void setLVUserEventPortInfo(unsigned long *lvUserEventRef);
01520 typedef void (__cdecl *SetLVUserEventPortInfoFuncPtr)(unsigned long *lvUserEventRef);
01521
01522
01526 #pragma pack(1)
01527 typedef struct lvDeviceStatusStruct
01528 {
01529     char portname[32];
01530     unsigned char devId;
01531     DeviceStatusTypes status;
01532     unsigned char devDataLen;
01533     unsigned char devData[255];
01534 } LabViewDeviceStatusType;
01535 #pragma pack()
01536
01541 NKTPDLL_EXPORT void setLVUserEventDeviceInfo(unsigned long *lvUserEventRef);
01542 typedef void (__cdecl *SetLVUserEventDeviceInfoFuncPtr)(unsigned long *lvUserEventRef);
01543
01547 #pragma pack(1)
01548 typedef struct lvRegisterStatusStruct
01549 {
01550     char portname[32];
01551     unsigned char devId;
01552     unsigned char regId;
01553     RegisterStatusTypes status;
01554     RegisterDataTypes regType;
01555     unsigned char regDataLen;
01556     unsigned char regData[255];
01557 } LabViewRegisterStatusType;
01558 #pragma pack()
01559
01564 NKTPDLL_EXPORT void setLVUserEventRegisterInfo(unsigned long *lvUserEventRef);
01565 typedef void (__cdecl *SetLVUserEventRegisterInfoFuncPtr)(unsigned long *lvUserEventRef);
01566
01569 #ifdef __cplusplus
01570 } // namespace NKTPDLL
01571 #endif
01572
01573 #endif // NKTPDLL_H

```

Index

closePorts
 NKTPDLL.h, [48](#)

ClosePortsFuncPtr
 NKTPDLL.h, [28](#)

curScanAdr
 lvPortStatusStruct, [9](#)

DateTimeType
 NKTPDLL.h, [27](#)

Day
 tDateTimeStruct, [12](#)

Denominator
 tParamSetStruct, [14](#)

Deprecated List, [1](#)

devData
 lvDeviceStatusStruct, [8](#)

devDataLen
 lvDeviceStatusStruct, [8](#)

DeviceBIVerChanged
 NKTPDLL.h, [43](#)

deviceCreate
 NKTPDLL.h, [80](#)

DeviceCreateFuncPtr
 NKTPDLL.h, [35](#)

DeviceErrorCodeChanged
 NKTPDLL.h, [43](#)

deviceExists
 NKTPDLL.h, [81](#)

DeviceExistsFuncPtr
 NKTPDLL.h, [35](#)

DeviceFwVerChanged
 NKTPDLL.h, [43](#)

deviceGetAllTypes
 NKTPDLL.h, [82](#)

DeviceGetAllTypesFuncPtr
 NKTPDLL.h, [35](#)

deviceGetAllTypesV2
 NKTPDLL.h, [83](#)

DeviceGetAllTypesV2FuncPtr
 NKTPDLL.h, [35](#)

deviceGetBootloaderVersion
 NKTPDLL.h, [77](#)

DeviceGetBootloaderVersionFuncPtr
 NKTPDLL.h, [34](#)

deviceGetBootloaderVersionStr
 NKTPDLL.h, [78](#)

DeviceGetBootloaderVersionStrFuncPtr
 NKTPDLL.h, [34](#)

deviceGetErrorCode
 NKTPDLL.h, [77](#)

DeviceGetErrorCodeFuncPtr
 NKTPDLL.h, [34](#)

deviceGetFirmwareVersion
 NKTPDLL.h, [78](#)

DeviceGetFirmwareVersionFuncPtr
 NKTPDLL.h, [34](#)

deviceGetFirmwareVersionStr
 NKTPDLL.h, [79](#)

DeviceGetFirmwareVersionStrFuncPtr
 NKTPDLL.h, [35](#)

deviceGetLive
 NKTPDLL.h, [84](#)

DeviceGetLiveFuncPtr
 NKTPDLL.h, [36](#)

deviceGetMode
 NKTPDLL.h, [84](#)

DeviceGetModeFuncPtr
 NKTPDLL.h, [36](#)

deviceGetModuleSerialNumberStr
 NKTPDLL.h, [79](#)

DeviceGetModuleSerialNumberStrFuncPtr
 NKTPDLL.h, [35](#)

deviceGetPartNumberStr
 NKTPDLL.h, [75](#)

DeviceGetPartNumberStrFuncPtr
 NKTPDLL.h, [34](#)

deviceGetPCBSerialNumberStr
 NKTPDLL.h, [80](#)

DeviceGetPCBSerialNumberStrFuncPtr
 NKTPDLL.h, [35](#)

deviceGetPCBVersion
 NKTPDLL.h, [76](#)

DeviceGetPCBVersionFuncPtr
 NKTPDLL.h, [34](#)

deviceGetStatusBits
 NKTPDLL.h, [76](#)

DeviceGetStatusBitsFuncPtr
 NKTPDLL.h, [34](#)

deviceGetSysType
 NKTPDLL.h, [75](#)

DeviceGetSysTypeFuncPtr
 NKTPDLL.h, [34](#)

deviceGetType
 NKTPDLL.h, [74](#)

DeviceGetTypeFuncPtr
 NKTPDLL.h, [33](#)

deviceGetTypeV2
 NKTPDLL.h, [74](#)

DeviceGetTypeV2FuncPtr

- NKTPDLL.h, 34
- DeviceLiveChanged
 - NKTPDLL.h, 43
- DeviceModeChanged
 - NKTPDLL.h, 43
- DeviceModeTypes
 - NKTPDLL.h, 27
- DeviceModuleSerialChanged
 - NKTPDLL.h, 43
- DevicePartNumberChanged
 - NKTPDLL.h, 43
- DevicePCBSerialChanged
 - NKTPDLL.h, 43
- DevicePCBVersionChanged
 - NKTPDLL.h, 43
- deviceRemove
 - NKTPDLL.h, 82
- deviceRemoveAll
 - NKTPDLL.h, 82
- DeviceRemoveAllFuncPtr
 - NKTPDLL.h, 35
- DeviceRemoveFuncPtr
 - NKTPDLL.h, 35
- DeviceResultTypes
 - NKTPDLL.h, 26
- deviceSetLive
 - NKTPDLL.h, 85
- DeviceSetLiveFuncPtr
 - NKTPDLL.h, 36
- DeviceStatusBitsChanged
 - NKTPDLL.h, 43
- DeviceStatusCallbackFuncPtr
 - NKTPDLL.h, 37
- DeviceStatusTypes
 - NKTPDLL.h, 27
- DeviceSysTypeChanged
 - NKTPDLL.h, 43
- DeviceTypeChanged
 - NKTPDLL.h, 43
- devId
 - lvDeviceStatusStruct, 7
 - lvRegisterStatusStruct, 10
- DevModeAnalyze
 - NKTPDLL.h, 40
- DevModeAnalyzeInit
 - NKTPDLL.h, 40
- DevModeDisabled
 - NKTPDLL.h, 40
- DevModeError
 - NKTPDLL.h, 40
- DevModeLogDownload
 - NKTPDLL.h, 40
- DevModeNormal
 - NKTPDLL.h, 40
- DevModeTimeout
 - NKTPDLL.h, 40
- DevModeUpload
 - NKTPDLL.h, 40
- DevResultApplicationBusy
 - NKTPDLL.h, 40
- DevResultDeviceNotFound
 - NKTPDLL.h, 40
- DevResultFailed
 - NKTPDLL.h, 40
- DevResultPortNotFound
 - NKTPDLL.h, 40
- DevResultPortOpenError
 - NKTPDLL.h, 40
- DevResultSuccess
 - NKTPDLL.h, 40
- DevResultWaitTimeout
 - NKTPDLL.h, 40
- ErrorHandler
 - tParamSetStruct, 13
- FactoryVal
 - tParamSetStruct, 14
- foundType
 - lvPortStatusStruct, 9
- getAllPorts
 - NKTPDLL.h, 45
- GetAllPortsFuncPtr
 - NKTPDLL.h, 28
- getLegacyBusScanning
 - NKTPDLL.h, 49
- GetLegacyBusScanningFuncPtr
 - NKTPDLL.h, 29
- getOpenPorts
 - NKTPDLL.h, 45
- GetOpenPortsFuncPtr
 - NKTPDLL.h, 28
- getPortErrorMsg
 - NKTPDLL.h, 50
- getPortErrorMsgFuncPtr
 - NKTPDLL.h, 29
- getPortStatus
 - NKTPDLL.h, 50
- getPortStatusFuncPtr
 - NKTPDLL.h, 29
- getSpecificBusScanningRange
 - NKTPDLL.h, 50
- GetSpecificBusScanningRangeFuncPtr
 - NKTPDLL.h, 29
- Hour
 - tDateTimeStruct, 12
- LabViewDeviceStatusType
 - NKTPDLL.h, 38
- LabViewPortStatusType
 - NKTPDLL.h, 38
- LabViewRegisterStatusType
 - NKTPDLL.h, 39
- LLimit
 - tParamSetStruct, 14

- lvDeviceStatusStruct, 7
 - devData, 8
 - devDataLen, 8
 - devId, 7
 - portname, 7
 - status, 8
- lvPortStatusStruct, 8
 - curScanAdr, 9
 - foundType, 9
 - maxScanAdr, 9
 - portname, 9
 - status, 9
- lvRegisterStatusStruct, 9
 - devId, 10
 - portname, 10
 - regData, 11
 - regDataLen, 11
 - regId, 10
 - regType, 10
 - status, 10
- maxScanAdr
 - lvPortStatusStruct, 9
- Min
 - tDateTimeStruct, 12
- Month
 - tDateTimeStruct, 12
- NKTPDLL.h, 15
 - closePorts, 48
 - ClosePortsFuncPtr, 28
 - DateTimeType, 27
 - DeviceBIVerChanged, 43
 - deviceCreate, 80
 - DeviceCreateFuncPtr, 35
 - DeviceErrorCodeChanged, 43
 - deviceExists, 81
 - DeviceExistsFuncPtr, 35
 - DeviceFwVerChanged, 43
 - deviceGetAllTypes, 82
 - DeviceGetAllTypesFuncPtr, 35
 - deviceGetAllTypesV2, 83
 - DeviceGetAllTypesV2FuncPtr, 35
 - deviceGetBootloaderVersion, 77
 - DeviceGetBootloaderVersionFuncPtr, 34
 - deviceGetBootloaderVersionStr, 78
 - DeviceGetBootloaderVersionStrFuncPtr, 34
 - deviceGetErrorCode, 77
 - DeviceGetErrorCodeFuncPtr, 34
 - deviceGetFirmwareVersion, 78
 - DeviceGetFirmwareVersionFuncPtr, 34
 - deviceGetFirmwareVersionStr, 79
 - DeviceGetFirmwareVersionStrFuncPtr, 35
 - deviceGetLive, 84
 - DeviceGetLiveFuncPtr, 36
 - deviceGetMode, 84
 - DeviceGetModeFuncPtr, 36
 - deviceGetModuleSerialNumberStr, 79
 - DeviceGetModuleSerialNumberStrFuncPtr, 35
 - deviceGetPartNumberStr, 75
 - DeviceGetPartNumberStrFuncPtr, 34
 - deviceGetPCBSerialNumberStr, 80
 - DeviceGetPCBSerialNumberStrFuncPtr, 35
 - deviceGetPCBVersion, 76
 - DeviceGetPCBVersionFuncPtr, 34
 - deviceGetStatusBits, 76
 - DeviceGetStatusBitsFuncPtr, 34
 - deviceGetSysType, 75
 - DeviceGetSysTypeFuncPtr, 34
 - deviceGetType, 74
 - DeviceGetTypeFuncPtr, 33
 - deviceGetTypeV2, 74
 - DeviceGetTypeV2FuncPtr, 34
 - DeviceLiveChanged, 43
 - DeviceModeChanged, 43
 - DeviceModeTypes, 27
 - DeviceModuleSerialChanged, 43
 - DevicePartNumberChanged, 43
 - DevicePCBSerialChanged, 43
 - DevicePCBVersionChanged, 43
 - deviceRemove, 82
 - deviceRemoveAll, 82
 - DeviceRemoveAllFuncPtr, 35
 - DeviceRemoveFuncPtr, 35
 - DeviceResultTypes, 26
 - deviceSetLive, 85
 - DeviceSetLiveFuncPtr, 36
 - DeviceStatusBitsChanged, 43
 - DeviceStatusCallbackFuncPtr, 37
 - DeviceStatusTypes, 27
 - DeviceSysTypeChanged, 43
 - DeviceTypeChanged, 43
 - DevModeAnalyze, 40
 - DevModeAnalyzeInit, 40
 - DevModeDisabled, 40
 - DevModeError, 40
 - DevModeLogDownload, 40
 - DevModeNormal, 40
 - DevModeTimeout, 40
 - DevModeUpload, 40
 - DevResultApplicationBusy, 40
 - DevResultDeviceNotFound, 40
 - DevResultFailed, 40
 - DevResultPortNotFound, 40
 - DevResultPortOpenError, 40
 - DevResultSuccess, 40
 - DevResultWaitTimeout, 40
 - getAllPorts, 45
 - GetAllPortsFuncPtr, 28
 - getLegacyBusScanning, 49
 - GetLegacyBusScanningFuncPtr, 29
 - getOpenPorts, 45
 - GetOpenPortsFuncPtr, 28
 - getPortErrorMsg, 50
 - getPortErrorMsgFuncPtr, 29
 - getPortStatus, 50
 - getPortStatusFuncPtr, 29

getSpecificBusScanningRange, 50
GetSpecificBusScanningRangeFuncPtr, 29
LabViewDeviceStatusType, 38
LabViewPortStatusType, 38
LabViewRegisterStatusType, 39
NKTPDLL_EXPORT, 26
OPApplicationBusy, 39
openPorts, 47
OpenPortsFuncPtr, 28
OPFailed, 39
OPNoDevices, 39
OPPortNotFound, 39
OPSuccess, 39
P2PApplicationBusy, 39
P2PInvalidLocalIP, 39
P2PInvalidPortname, 39
P2PInvalidRemoteIP, 39
P2PPortnameExists, 39
P2PPortnameNotFound, 39
P2PPortResultTypes, 26
P2PSuccess, 39
ParameterSetType, 27
ParamSetUnitTypes, 27
pointToPointPortAdd, 45
PointToPointPortAddFuncPtr, 28
pointToPointPortDel, 47
PointToPointPortDelFuncPtr, 28
pointToPointPortGet, 46
PointToPointPortGetFuncPtr, 28
PortClosed, 42
PortClosing, 42
PortOpened, 42
PortOpenFail, 42
PortOpening, 42
PortReady, 42
PortResultTypes, 26
PortScanDeviceFound, 42
PortScanEnded, 42
PortScanProgress, 42
PortScanStarted, 42
PortStatusCallbackFuncPtr, 36
PortStatusTypes, 27
PortStatusUnknown, 42
RegBusy, 43
RegComError, 43
RegCRCError, 43
RegData_Ascii, 41
RegData_B16, 41
RegData_B32, 42
RegData_B64, 42
RegData_B8, 41
RegData_DateTime, 42
RegData_F32, 41
RegData_F64, 41
RegData_H16, 42
RegData_H32, 42
RegData_H64, 42
RegData_H8, 41
RegData_Mixed, 41
RegData_Paramset, 41
RegData_S16, 41
RegData_S32, 41
RegData_S64, 41
RegData_S8, 41
RegData_U16, 41
RegData_U32, 41
RegData_U64, 41
RegData_U8, 41
RegData_Unknown, 41
registerCreate, 85
RegisterCreateFuncPtr, 36
RegisterDataTypes, 27
registerExists, 86
RegisterExistsFuncPtr, 36
registerGetAll, 87
RegisterGetAllFuncPtr, 36
RegisterPriorityTypes, 27
registerRead, 51
registerReadAscii, 57
RegisterReadAsciiFuncPtr, 30
registerReadF32, 56
RegisterReadF32FuncPtr, 30
registerReadF64, 57
RegisterReadF64FuncPtr, 30
RegisterReadFuncPtr, 29
registerReadS16, 53
RegisterReadS16FuncPtr, 30
registerReadS32, 54
RegisterReadS32FuncPtr, 30
registerReadS64, 56
RegisterReadS64FuncPtr, 30
registerReadS8, 52
RegisterReadS8FuncPtr, 29
registerReadU16, 53
RegisterReadU16FuncPtr, 29
registerReadU32, 54
RegisterReadU32FuncPtr, 30
registerReadU64, 55
RegisterReadU64FuncPtr, 30
registerReadU8, 51
RegisterReadU8FuncPtr, 29
registerRemove, 86
registerRemoveAll, 87
RegisterRemoveAllFuncPtr, 36
RegisterRemoveFuncPtr, 36
RegisterResultTypes, 27
RegisterStatusCallbackFuncPtr, 37
RegisterStatusTypes, 27
registerWrite, 58
registerWriteAscii, 65
RegisterWriteAsciiFuncPtr, 32
registerWriteF32, 64
RegisterWriteF32FuncPtr, 32
registerWriteF64, 64
RegisterWriteF64FuncPtr, 32
RegisterWriteFuncPtr, 30

registerWriteRead, 66
registerWriteReadAscii, 73
RegisterWriteReadAsciiFuncPtr, 33
registerWriteReadF32, 72
RegisterWriteReadF32FuncPtr, 33
registerWriteReadF64, 72
RegisterWriteReadF64FuncPtr, 33
RegisterWriteReadFuncPtr, 32
registerWriteReadS16, 68
RegisterWriteReadS16FuncPtr, 32
registerWriteReadS32, 70
RegisterWriteReadS32FuncPtr, 33
registerWriteReadS64, 71
RegisterWriteReadS64FuncPtr, 33
registerWriteReadS8, 67
RegisterWriteReadS8FuncPtr, 32
registerWriteReadU16, 68
RegisterWriteReadU16FuncPtr, 32
registerWriteReadU32, 69
RegisterWriteReadU32FuncPtr, 33
registerWriteReadU64, 70
RegisterWriteReadU64FuncPtr, 33
registerWriteReadU8, 66
RegisterWriteReadU8FuncPtr, 32
registerWriteS16, 61
RegisterWriteS16FuncPtr, 31
registerWriteS32, 62
RegisterWriteS32FuncPtr, 31
registerWriteS64, 63
RegisterWriteS64FuncPtr, 31
registerWriteS8, 59
RegisterWriteS8FuncPtr, 31
registerWriteU16, 60
RegisterWriteU16FuncPtr, 31
registerWriteU32, 61
RegisterWriteU32FuncPtr, 31
registerWriteU64, 62
RegisterWriteU64FuncPtr, 31
registerWriteU8, 59
RegisterWriteU8FuncPtr, 31
RegNacked, 43
RegPriority_High, 42
RegPriority_Low, 42
RegResultApplicationBusy, 41
RegResultBusy, 41
RegResultComError, 41
RegResultCRCError, 41
RegResultDeviceNotFound, 41
RegResultFailed, 40
RegResultIndexError, 41
RegResultNacked, 41
RegResultPortClosed, 41
RegResultPortNotFound, 41
RegResultPortOpenError, 41
RegResultReadError, 40
RegResultRegisterNotFound, 41
RegResultSuccess, 40
RegResultTimeout, 41
RegResultTypeError, 41
RegSuccess, 43
RegTimeout, 43
setCallbackPtrDeviceInfo, 88
SetCallbackPtrDeviceInfoFuncPtr, 37
setCallbackPtrPortInfo, 88
SetCallbackPtrPortInfoFuncPtr, 37
setCallbackPtrRegisterInfo, 88
SetCallbackPtrRegisterInfoFuncPtr, 38
setLegacyBusScanning, 48
SetLegacyBusScanningFuncPtr, 28
setLVUserEventDeviceInfo, 89
SetLVUserEventDeviceInfoFuncPtr, 38
setLVUserEventPortInfo, 88
SetLVUserEventPortInfoFuncPtr, 38
setLVUserEventRegisterInfo, 89
SetLVUserEventRegisterInfoFuncPtr, 39
setSpecificBusScanningRange, 49
SetSpecificBusScanningRangeFuncPtr, 29
tDeviceModeTypes, 40
tDeviceResultTypes, 40
tDeviceStatusTypes, 42
tP2PPortResultTypes, 39
tParamSetUnitTypes, 43
tPortResultTypes, 39
tPortStatusTypes, 42
tRegisterDataTypes, 41
tRegisterPriorityTypes, 42
tRegisterResultTypes, 40
tRegisterStatusTypes, 43
UnitA, 44
UnitcA, 44
UnitcB, 44
UnitcBm, 44
UnitcC, 44
UnitcIm, 44
UnitcmA, 44
UnitcmW, 44
UnitcuA, 44
UnitcV, 44
UnitcW, 44
UnitdA, 44
UnitdB, 44
UnitdBm, 44
UnitdC, 44
UnitdIm, 44
UnitdmA, 44
UnitdmW, 44
Unitdnm, 44
Unitdpm, 44
UnitduA, 44
UnitdV, 44
UnitdW, 44
UnitHz, 44
UnitkHz, 44
UnitIm, 44
UnitmA, 44
UnitmB, 44

- UnitmBm, [44](#)
- UnitmC, [44](#)
- UnitMHz, [44](#)
- Unitlm, [44](#)
- UnitmSec, [44](#)
- UnitmV, [43](#)
- UnitmW, [44](#)
- UnitnA, [44](#)
- Unitnm, [44](#)
- UnitNone, [43](#)
- UnitpA, [44](#)
- UnitPerCent, [44](#)
- UnitPerMille, [44](#)
- Unitpm, [44](#)
- UnitRPM, [44](#)
- UnitSec, [44](#)
- UnituA, [44](#)
- UnituSec, [44](#)
- UnituW, [44](#)
- UnitV, [43](#)
- UnitW, [44](#)
- NKTPDLL_EXPORT
 - NKTPDLL.h, [26](#)
- Numerator
 - tParamSetStruct, [14](#)
- Offset
 - tParamSetStruct, [14](#)
- OPApplicationBusy
 - NKTPDLL.h, [39](#)
- openPorts
 - NKTPDLL.h, [47](#)
- OpenPortsFuncPtr
 - NKTPDLL.h, [28](#)
- OPFailed
 - NKTPDLL.h, [39](#)
- OPNoDevices
 - NKTPDLL.h, [39](#)
- OPPortNotFound
 - NKTPDLL.h, [39](#)
- OPSuccess
 - NKTPDLL.h, [39](#)
- P2PApplicationBusy
 - NKTPDLL.h, [39](#)
- P2PInvalidLocalIP
 - NKTPDLL.h, [39](#)
- P2PInvalidPortname
 - NKTPDLL.h, [39](#)
- P2PInvalidRemoteIP
 - NKTPDLL.h, [39](#)
- P2PPortnameExists
 - NKTPDLL.h, [39](#)
- P2PPortnameNotFound
 - NKTPDLL.h, [39](#)
- P2PPortResultTypes
 - NKTPDLL.h, [26](#)
- P2PSuccess
 - NKTPDLL.h, [39](#)
- ParameterSetType
 - NKTPDLL.h, [27](#)
- ParamSetUnitTypes
 - NKTPDLL.h, [27](#)
- pointToPointPortAdd
 - NKTPDLL.h, [45](#)
- PointToPointPortAddFuncPtr
 - NKTPDLL.h, [28](#)
- pointToPointPortDel
 - NKTPDLL.h, [47](#)
- PointToPointPortDelFuncPtr
 - NKTPDLL.h, [28](#)
- pointToPointPortGet
 - NKTPDLL.h, [46](#)
- PointToPointPortGetFuncPtr
 - NKTPDLL.h, [28](#)
- PortClosed
 - NKTPDLL.h, [42](#)
- PortClosing
 - NKTPDLL.h, [42](#)
- portname
 - lvDeviceStatusStruct, [7](#)
 - lvPortStatusStruct, [9](#)
 - lvRegisterStatusStruct, [10](#)
- PortOpened
 - NKTPDLL.h, [42](#)
- PortOpenFail
 - NKTPDLL.h, [42](#)
- PortOpening
 - NKTPDLL.h, [42](#)
- PortReady
 - NKTPDLL.h, [42](#)
- PortResultTypes
 - NKTPDLL.h, [26](#)
- PortScanDeviceFound
 - NKTPDLL.h, [42](#)
- PortScanEnded
 - NKTPDLL.h, [42](#)
- PortScanProgress
 - NKTPDLL.h, [42](#)
- PortScanStarted
 - NKTPDLL.h, [42](#)
- PortStatusCallbackFuncPtr
 - NKTPDLL.h, [36](#)
- PortStatusTypes
 - NKTPDLL.h, [27](#)
- PortStatusUnknown
 - NKTPDLL.h, [42](#)
- RegBusy
 - NKTPDLL.h, [43](#)
- RegComError
 - NKTPDLL.h, [43](#)
- RegCRCErr
 - NKTPDLL.h, [43](#)
- regData
 - lvRegisterStatusStruct, [11](#)
- RegData_Ascii
 - NKTPDLL.h, [41](#)

RegData_B16
 NKTPDLL.h, [41](#)
RegData_B32
 NKTPDLL.h, [42](#)
RegData_B64
 NKTPDLL.h, [42](#)
RegData_B8
 NKTPDLL.h, [41](#)
RegData_DateTime
 NKTPDLL.h, [42](#)
RegData_F32
 NKTPDLL.h, [41](#)
RegData_F64
 NKTPDLL.h, [41](#)
RegData_H16
 NKTPDLL.h, [42](#)
RegData_H32
 NKTPDLL.h, [42](#)
RegData_H64
 NKTPDLL.h, [42](#)
RegData_H8
 NKTPDLL.h, [41](#)
RegData_Mixed
 NKTPDLL.h, [41](#)
RegData_Paramset
 NKTPDLL.h, [41](#)
RegData_S16
 NKTPDLL.h, [41](#)
RegData_S32
 NKTPDLL.h, [41](#)
RegData_S64
 NKTPDLL.h, [41](#)
RegData_S8
 NKTPDLL.h, [41](#)
RegData_U16
 NKTPDLL.h, [41](#)
RegData_U32
 NKTPDLL.h, [41](#)
RegData_U64
 NKTPDLL.h, [41](#)
RegData_U8
 NKTPDLL.h, [41](#)
RegData_Unknown
 NKTPDLL.h, [41](#)
regDataLen
 lvRegisterStatusStruct, [11](#)
regId
 lvRegisterStatusStruct, [10](#)
registerCreate
 NKTPDLL.h, [85](#)
RegisterCreateFuncPtr
 NKTPDLL.h, [36](#)
RegisterDataTypes
 NKTPDLL.h, [27](#)
registerExists
 NKTPDLL.h, [86](#)
RegisterExistsFuncPtr
 NKTPDLL.h, [36](#)
registerGetAll
 NKTPDLL.h, [87](#)
RegisterGetAllFuncPtr
 NKTPDLL.h, [36](#)
RegisterPriorityTypes
 NKTPDLL.h, [27](#)
registerRead
 NKTPDLL.h, [51](#)
registerReadAscii
 NKTPDLL.h, [57](#)
RegisterReadAsciiFuncPtr
 NKTPDLL.h, [30](#)
registerReadF32
 NKTPDLL.h, [56](#)
RegisterReadF32FuncPtr
 NKTPDLL.h, [30](#)
registerReadF64
 NKTPDLL.h, [57](#)
RegisterReadF64FuncPtr
 NKTPDLL.h, [30](#)
RegisterReadFuncPtr
 NKTPDLL.h, [29](#)
registerReadS16
 NKTPDLL.h, [53](#)
RegisterReadS16FuncPtr
 NKTPDLL.h, [30](#)
registerReadS32
 NKTPDLL.h, [54](#)
RegisterReadS32FuncPtr
 NKTPDLL.h, [30](#)
registerReadS64
 NKTPDLL.h, [56](#)
RegisterReadS64FuncPtr
 NKTPDLL.h, [30](#)
registerReadS8
 NKTPDLL.h, [52](#)
RegisterReadS8FuncPtr
 NKTPDLL.h, [29](#)
registerReadU16
 NKTPDLL.h, [53](#)
RegisterReadU16FuncPtr
 NKTPDLL.h, [29](#)
registerReadU32
 NKTPDLL.h, [54](#)
RegisterReadU32FuncPtr
 NKTPDLL.h, [30](#)
registerReadU64
 NKTPDLL.h, [55](#)
RegisterReadU64FuncPtr
 NKTPDLL.h, [30](#)
registerReadU8
 NKTPDLL.h, [51](#)
RegisterReadU8FuncPtr
 NKTPDLL.h, [29](#)
registerRemove
 NKTPDLL.h, [86](#)
registerRemoveAll
 NKTPDLL.h, [87](#)

RegisterRemoveAllFuncPtr
NKTPDLL.h, [36](#)

RegisterRemoveFuncPtr
NKTPDLL.h, [36](#)

RegisterResultTypes
NKTPDLL.h, [27](#)

RegisterStatusCallbackFuncPtr
NKTPDLL.h, [37](#)

RegisterStatusTypes
NKTPDLL.h, [27](#)

registerWrite
NKTPDLL.h, [58](#)

registerWriteAscii
NKTPDLL.h, [65](#)

RegisterWriteAsciiFuncPtr
NKTPDLL.h, [32](#)

registerWriteF32
NKTPDLL.h, [64](#)

RegisterWriteF32FuncPtr
NKTPDLL.h, [32](#)

registerWriteF64
NKTPDLL.h, [64](#)

RegisterWriteF64FuncPtr
NKTPDLL.h, [32](#)

RegisterWriteFuncPtr
NKTPDLL.h, [30](#)

registerWriteRead
NKTPDLL.h, [66](#)

registerWriteReadAscii
NKTPDLL.h, [73](#)

RegisterWriteReadAsciiFuncPtr
NKTPDLL.h, [33](#)

registerWriteReadF32
NKTPDLL.h, [72](#)

RegisterWriteReadF32FuncPtr
NKTPDLL.h, [33](#)

registerWriteReadF64
NKTPDLL.h, [72](#)

RegisterWriteReadF64FuncPtr
NKTPDLL.h, [33](#)

RegisterWriteReadFuncPtr
NKTPDLL.h, [32](#)

registerWriteReadS16
NKTPDLL.h, [68](#)

RegisterWriteReadS16FuncPtr
NKTPDLL.h, [32](#)

registerWriteReadS32
NKTPDLL.h, [70](#)

RegisterWriteReadS32FuncPtr
NKTPDLL.h, [33](#)

registerWriteReadS64
NKTPDLL.h, [71](#)

RegisterWriteReadS64FuncPtr
NKTPDLL.h, [33](#)

registerWriteReadS8
NKTPDLL.h, [67](#)

RegisterWriteReadS8FuncPtr
NKTPDLL.h, [32](#)

registerWriteReadU16
NKTPDLL.h, [68](#)

RegisterWriteReadU16FuncPtr
NKTPDLL.h, [32](#)

registerWriteReadU32
NKTPDLL.h, [69](#)

RegisterWriteReadU32FuncPtr
NKTPDLL.h, [33](#)

registerWriteReadU64
NKTPDLL.h, [70](#)

RegisterWriteReadU64FuncPtr
NKTPDLL.h, [33](#)

registerWriteReadU8
NKTPDLL.h, [66](#)

RegisterWriteReadU8FuncPtr
NKTPDLL.h, [32](#)

registerWriteS16
NKTPDLL.h, [61](#)

RegisterWriteS16FuncPtr
NKTPDLL.h, [31](#)

registerWriteS32
NKTPDLL.h, [62](#)

RegisterWriteS32FuncPtr
NKTPDLL.h, [31](#)

registerWriteS64
NKTPDLL.h, [63](#)

RegisterWriteS64FuncPtr
NKTPDLL.h, [31](#)

registerWriteS8
NKTPDLL.h, [59](#)

RegisterWriteS8FuncPtr
NKTPDLL.h, [31](#)

registerWriteU16
NKTPDLL.h, [60](#)

RegisterWriteU16FuncPtr
NKTPDLL.h, [31](#)

registerWriteU32
NKTPDLL.h, [61](#)

RegisterWriteU32FuncPtr
NKTPDLL.h, [31](#)

registerWriteU64
NKTPDLL.h, [62](#)

RegisterWriteU64FuncPtr
NKTPDLL.h, [31](#)

registerWriteU8
NKTPDLL.h, [59](#)

RegisterWriteU8FuncPtr
NKTPDLL.h, [31](#)

RegNacked
NKTPDLL.h, [43](#)

RegPriority_High
NKTPDLL.h, [42](#)

RegPriority_Low
NKTPDLL.h, [42](#)

RegResultApplicationBusy
NKTPDLL.h, [41](#)

RegResultBusy
NKTPDLL.h, [41](#)

- RegResultComError
 - NKTPDLL.h, [41](#)
- RegResultCRCError
 - NKTPDLL.h, [41](#)
- RegResultDeviceNotFound
 - NKTPDLL.h, [41](#)
- RegResultFailed
 - NKTPDLL.h, [40](#)
- RegResultIndexError
 - NKTPDLL.h, [41](#)
- RegResultNacked
 - NKTPDLL.h, [41](#)
- RegResultPortClosed
 - NKTPDLL.h, [41](#)
- RegResultPortNotFound
 - NKTPDLL.h, [41](#)
- RegResultPortOpenError
 - NKTPDLL.h, [41](#)
- RegResultReadError
 - NKTPDLL.h, [40](#)
- RegResultRegisterNotFound
 - NKTPDLL.h, [41](#)
- RegResultSuccess
 - NKTPDLL.h, [40](#)
- RegResultTimeout
 - NKTPDLL.h, [41](#)
- RegResultTypeError
 - NKTPDLL.h, [41](#)
- RegSuccess
 - NKTPDLL.h, [43](#)
- RegTimeout
 - NKTPDLL.h, [43](#)
- regType
 - lvRegisterStatusStruct, [10](#)
- Sec
 - tDateTimeStruct, [12](#)
- setCallbackPtrDeviceInfo
 - NKTPDLL.h, [88](#)
- SetCallbackPtrDeviceInfoFuncPtr
 - NKTPDLL.h, [37](#)
- setCallbackPtrPortInfo
 - NKTPDLL.h, [88](#)
- SetCallbackPtrPortInfoFuncPtr
 - NKTPDLL.h, [37](#)
- setCallbackPtrRegisterInfo
 - NKTPDLL.h, [88](#)
- SetCallbackPtrRegisterInfoFuncPtr
 - NKTPDLL.h, [38](#)
- setLegacyBusScanning
 - NKTPDLL.h, [48](#)
- SetLegacyBusScanningFuncPtr
 - NKTPDLL.h, [28](#)
- setLVUserEventDeviceInfo
 - NKTPDLL.h, [89](#)
- SetLVUserEventDeviceInfoFuncPtr
 - NKTPDLL.h, [38](#)
- setLVUserEventPortInfo
 - NKTPDLL.h, [88](#)
- SetLVUserEventPortInfoFuncPtr
 - NKTPDLL.h, [38](#)
- setLVUserEventRegisterInfo
 - NKTPDLL.h, [89](#)
- SetLVUserEventRegisterInfoFuncPtr
 - NKTPDLL.h, [39](#)
- setSpecificBusScanningRange
 - NKTPDLL.h, [49](#)
- SetSpecificBusScanningRangeFuncPtr
 - NKTPDLL.h, [29](#)
- StartVal
 - tParamSetStruct, [13](#)
- status
 - lvDeviceStatusStruct, [8](#)
 - lvPortStatusStruct, [9](#)
 - lvRegisterStatusStruct, [10](#)
- tDateTimeStruct, [11](#)
 - Day, [12](#)
 - Hour, [12](#)
 - Min, [12](#)
 - Month, [12](#)
 - Sec, [12](#)
 - Year, [12](#)
- tDeviceModeTypes
 - NKTPDLL.h, [40](#)
- tDeviceResultTypes
 - NKTPDLL.h, [40](#)
- tDeviceStatusTypes
 - NKTPDLL.h, [42](#)
- tP2PPortResultTypes
 - NKTPDLL.h, [39](#)
- tParamSetStruct, [12](#)
 - Denominator, [14](#)
 - ErrorHandler, [13](#)
 - FactoryVal, [14](#)
 - LLimit, [14](#)
 - Numerator, [14](#)
 - Offset, [14](#)
 - StartVal, [13](#)
 - ULimit, [14](#)
 - Unit, [13](#)
- tParamSetUnitTypes
 - NKTPDLL.h, [43](#)
- tPortResultTypes
 - NKTPDLL.h, [39](#)
- tPortStatusTypes
 - NKTPDLL.h, [42](#)
- tRegisterDataTypes
 - NKTPDLL.h, [41](#)
- tRegisterPriorityTypes
 - NKTPDLL.h, [42](#)
- tRegisterResultTypes
 - NKTPDLL.h, [40](#)
- tRegisterStatusTypes
 - NKTPDLL.h, [43](#)
- ULimit
 - tParamSetStruct, [14](#)

Unit
 tParamSetStruct, 13
UnitA
 NKTPDLL.h, 44
UnitcA
 NKTPDLL.h, 44
UnitcB
 NKTPDLL.h, 44
UnitcBm
 NKTPDLL.h, 44
UnitcC
 NKTPDLL.h, 44
Unitclm
 NKTPDLL.h, 44
UnitcmA
 NKTPDLL.h, 44
UnitcmW
 NKTPDLL.h, 44
UnitcuA
 NKTPDLL.h, 44
UnitcV
 NKTPDLL.h, 44
UnitcW
 NKTPDLL.h, 44
UnitdA
 NKTPDLL.h, 44
UnitdB
 NKTPDLL.h, 44
UnitdBm
 NKTPDLL.h, 44
UnitdC
 NKTPDLL.h, 44
Unitdlm
 NKTPDLL.h, 44
UnitdmA
 NKTPDLL.h, 44
UnitdmW
 NKTPDLL.h, 44
Unitdnm
 NKTPDLL.h, 44
Unitdpm
 NKTPDLL.h, 44
UnitduA
 NKTPDLL.h, 44
UnitdV
 NKTPDLL.h, 44
UnitdW
 NKTPDLL.h, 44
UnitHz
 NKTPDLL.h, 44
UnitkHz
 NKTPDLL.h, 44
Unitlm
 NKTPDLL.h, 44
UnitmA
 NKTPDLL.h, 44
UnitmB
 NKTPDLL.h, 44
UnitmBm
 NKTPDLL.h, 44
UnitmC
 NKTPDLL.h, 44
UnitMHz
 NKTPDLL.h, 44
Unitlm
 NKTPDLL.h, 44
UnitmSec
 NKTPDLL.h, 44
UnitmV
 NKTPDLL.h, 43
UnitmW
 NKTPDLL.h, 44
UnitnA
 NKTPDLL.h, 44
Unitnm
 NKTPDLL.h, 44
UnitNone
 NKTPDLL.h, 43
UnitpA
 NKTPDLL.h, 44
UnitPerCent
 NKTPDLL.h, 44
UnitPerMille
 NKTPDLL.h, 44
Unitpm
 NKTPDLL.h, 44
UnitRPM
 NKTPDLL.h, 44
UnitSec
 NKTPDLL.h, 44
UnituA
 NKTPDLL.h, 44
UnituSec
 NKTPDLL.h, 44
UnituW
 NKTPDLL.h, 44
UnitV
 NKTPDLL.h, 43
UnitW
 NKTPDLL.h, 44
Year
 tDateTimeStruct, 12