

# NKTPDLL Reference manual

Generated by Doxygen 1.8.12



# Contents

- 1 Data Structure Index** **1**
- 1.1 Data Structures . . . . . 1
- 2 File Index** **3**
- 2.1 File List . . . . . 3
- 3 Data Structure Documentation** **5**
- 3.1 IvDeviceStatusStruct Struct Reference . . . . . 5
- 3.1.1 Detailed Description . . . . . 5
- 3.1.2 Field Documentation . . . . . 5
- 3.1.2.1 portname . . . . . 5
- 3.1.2.2 devId . . . . . 6
- 3.1.2.3 status . . . . . 6
- 3.1.2.4 devDataLen . . . . . 6
- 3.1.2.5 devData . . . . . 6
- 3.2 IvPortStatusStruct Struct Reference . . . . . 6
- 3.2.1 Detailed Description . . . . . 6
- 3.2.2 Field Documentation . . . . . 7
- 3.2.2.1 portname . . . . . 7
- 3.2.2.2 status . . . . . 7
- 3.2.2.3 curScanAdr . . . . . 7
- 3.2.2.4 maxScanAdr . . . . . 7
- 3.2.2.5 foundType . . . . . 7
- 3.3 IvRegisterStatusStruct Struct Reference . . . . . 7

---

3.3.1	Detailed Description	8
3.3.2	Field Documentation	8
3.3.2.1	portname	8
3.3.2.2	devId	8
3.3.2.3	regId	8
3.3.2.4	status	8
3.3.2.5	regType	9
3.3.2.6	regDataLen	9
3.3.2.7	regData	9
3.4	tDateTimeStruct Struct Reference	9
3.4.1	Detailed Description	9
3.4.2	Field Documentation	10
3.4.2.1	Sec	10
3.4.2.2	Min	10
3.4.2.3	Hour	10
3.4.2.4	Day	10
3.4.2.5	Month	10
3.4.2.6	Year	10
3.5	tParamSetStruct Struct Reference	10
3.5.1	Detailed Description	11
3.5.2	Field Documentation	11
3.5.2.1	Unit	11
3.5.2.2	ErrorHandler	11
3.5.2.3	StartVal	12
3.5.2.4	FactoryVal	12
3.5.2.5	ULimit	12
3.5.2.6	LLimit	12
3.5.2.7	Numerator	12
3.5.2.8	Denominator	12
3.5.2.9	Offset	12

---

<b>4 File Documentation</b>	<b>13</b>
4.1 NKTPDLL.h File Reference	13
4.1.1 Detailed Description	23
4.1.2 Macro Definition Documentation	24
4.1.2.1 NKTPDLL_EXPORT	24
4.1.3 Typedef Documentation	24
4.1.3.1 PortResultTypes	24
4.1.3.2 P2PPortResultTypes	24
4.1.3.3 DeviceResultTypes	24
4.1.3.4 DeviceModeTypes	24
4.1.3.5 RegisterResultTypes	24
4.1.3.6 RegisterDataTypes	24
4.1.3.7 RegisterPriorityTypes	24
4.1.3.8 PortStatusTypes	25
4.1.3.9 DeviceStatusTypes	25
4.1.3.10 RegisterStatusTypes	25
4.1.3.11 DateTimeType	25
4.1.3.12 ParamSetUnitTypes	25
4.1.3.13 ParameterSetType	25
4.1.3.14 GetAllPortsFuncPtr	25
4.1.3.15 GetOpenPortsFuncPtr	25
4.1.3.16 PointToPointPortAddFuncPtr	26
4.1.3.17 PointToPointPortGetFuncPtr	26
4.1.3.18 PointToPointPortDelFuncPtr	26
4.1.3.19 OpenPortsFuncPtr	26
4.1.3.20 ClosePortsFuncPtr	26
4.1.3.21 SetLegacyBusScanningFuncPtr	26
4.1.3.22 GetLegacyBusScanningFuncPtr	26
4.1.3.23 getPortStatusFuncPtr	26
4.1.3.24 getPortErrorMsgFuncPtr	26

---

4.1.3.25	RegisterReadFuncPtr	27
4.1.3.26	RegisterReadU8FuncPtr	27
4.1.3.27	RegisterReadS8FuncPtr	27
4.1.3.28	RegisterReadU16FuncPtr	27
4.1.3.29	RegisterReadS16FuncPtr	27
4.1.3.30	RegisterReadU32FuncPtr	27
4.1.3.31	RegisterReadS32FuncPtr	27
4.1.3.32	RegisterReadU64FuncPtr	27
4.1.3.33	RegisterReadS64FuncPtr	27
4.1.3.34	RegisterReadF32FuncPtr	28
4.1.3.35	RegisterReadF64FuncPtr	28
4.1.3.36	RegisterReadAsciiFuncPtr	28
4.1.3.37	RegisterWriteFuncPtr	28
4.1.3.38	RegisterWriteU8FuncPtr	28
4.1.3.39	RegisterWriteS8FuncPtr	28
4.1.3.40	RegisterWriteU16FuncPtr	28
4.1.3.41	RegisterWriteS16FuncPtr	28
4.1.3.42	RegisterWriteU32FuncPtr	29
4.1.3.43	RegisterWriteS32FuncPtr	29
4.1.3.44	RegisterWriteU64FuncPtr	29
4.1.3.45	RegisterWriteS64FuncPtr	29
4.1.3.46	RegisterWriteF32FuncPtr	29
4.1.3.47	RegisterWriteF64FuncPtr	29
4.1.3.48	RegisterWriteAsciiFuncPtr	29
4.1.3.49	RegisterWriteReadFuncPtr	29
4.1.3.50	RegisterWriteReadU8FuncPtr	30
4.1.3.51	RegisterWriteReadS8FuncPtr	30
4.1.3.52	RegisterWriteReadU16FuncPtr	30
4.1.3.53	RegisterWriteReadS16FuncPtr	30
4.1.3.54	RegisterWriteReadU32FuncPtr	30

---

---

4.1.3.55	RegisterWriteReadS32FuncPtr	30
4.1.3.56	RegisterWriteReadU64FuncPtr	30
4.1.3.57	RegisterWriteReadS64FuncPtr	30
4.1.3.58	RegisterWriteReadF32FuncPtr	31
4.1.3.59	RegisterWriteReadF64FuncPtr	31
4.1.3.60	RegisterWriteReadAsciiFuncPtr	31
4.1.3.61	DeviceGetTypeFuncPtr	31
4.1.3.62	DeviceGetSysTypeFuncPtr	31
4.1.3.63	DeviceGetPartNumberStrFuncPtr	31
4.1.3.64	DeviceGetPCBVersionFuncPtr	31
4.1.3.65	DeviceGetStatusBitsFuncPtr	31
4.1.3.66	DeviceGetErrorCodeFuncPtr	32
4.1.3.67	DeviceGetBootloaderVersionFuncPtr	32
4.1.3.68	DeviceGetBootloaderVersionStrFuncPtr	32
4.1.3.69	DeviceGetFirmwareVersionFuncPtr	32
4.1.3.70	DeviceGetFirmwareVersionStrFuncPtr	32
4.1.3.71	DeviceGetModuleSerialNumberStrFuncPtr	32
4.1.3.72	DeviceGetPCBSerialNumberStrFuncPtr	32
4.1.3.73	DeviceCreateFuncPtr	32
4.1.3.74	DeviceExistsFuncPtr	32
4.1.3.75	DeviceRemoveFuncPtr	33
4.1.3.76	DeviceRemoveAllFuncPtr	33
4.1.3.77	DeviceGetAllTypesFuncPtr	33
4.1.3.78	DeviceGetModeFuncPtr	33
4.1.3.79	DeviceGetLiveFuncPtr	33
4.1.3.80	DeviceSetLiveFuncPtr	33
4.1.3.81	RegisterCreateFuncPtr	33
4.1.3.82	RegisterExistsFuncPtr	33
4.1.3.83	RegisterRemoveFuncPtr	33
4.1.3.84	RegisterRemoveAllFuncPtr	34

---

4.1.3.85	RegisterGetAllFuncPtr	34
4.1.3.86	PortStatusCallbackFuncPtr	34
4.1.3.87	SetCallbackPtrPortInfoFuncPtr	34
4.1.3.88	DeviceStatusCallbackFuncPtr	34
4.1.3.89	SetCallbackPtrDeviceInfoFuncPtr	35
4.1.3.90	RegisterStatusCallbackFuncPtr	35
4.1.3.91	SetCallbackPtrRegisterInfoFuncPtr	35
4.1.3.92	LabViewPortStatusType	35
4.1.3.93	SetLVUserEventPortInfoFuncPtr	36
4.1.3.94	LabViewDeviceStatusType	36
4.1.3.95	SetLVUserEventDeviceInfoFuncPtr	36
4.1.3.96	LabViewRegisterStatusType	36
4.1.3.97	SetLVUserEventRegisterInfoFuncPtr	36
4.1.4	Enumeration Type Documentation	36
4.1.4.1	tPortResultTypes	36
4.1.4.2	tP2PPortResultTypes	36
4.1.4.3	tDeviceResultTypes	38
4.1.4.4	tDeviceModeTypes	38
4.1.4.5	tRegisterResultTypes	39
4.1.4.6	tRegisterDataTypes	39
4.1.4.7	tRegisterPriorityTypes	40
4.1.4.8	tPortStatusTypes	40
4.1.4.9	tDeviceStatusTypes	41
4.1.4.10	tRegisterStatusTypes	41
4.1.4.11	tParamSetUnitTypes	41
4.1.5	Function Documentation	42
4.1.5.1	getAllPorts()	42
4.1.5.2	getOpenPorts()	43
4.1.5.3	pointToPointPortAdd()	43
4.1.5.4	pointToPointPortGet()	43

---

4.1.5.5	pointToPointPortDel()	44
4.1.5.6	openPorts()	45
4.1.5.7	closePorts()	45
4.1.5.8	setLegacyBusScanning()	46
4.1.5.9	getLegacyBusScanning()	46
4.1.5.10	getPortStatus()	46
4.1.5.11	getPortErrorMsg()	47
4.1.5.12	registerRead()	47
4.1.5.13	registerReadU8()	48
4.1.5.14	registerReadS8()	49
4.1.5.15	registerReadU16()	49
4.1.5.16	registerReadS16()	50
4.1.5.17	registerReadU32()	50
4.1.5.18	registerReadS32()	51
4.1.5.19	registerReadU64()	52
4.1.5.20	registerReadS64()	52
4.1.5.21	registerReadF32()	53
4.1.5.22	registerReadF64()	53
4.1.5.23	registerReadAscii()	54
4.1.5.24	registerWrite()	55
4.1.5.25	registerWriteU8()	55
4.1.5.26	registerWriteS8()	56
4.1.5.27	registerWriteU16()	57
4.1.5.28	registerWriteS16()	57
4.1.5.29	registerWriteU32()	58
4.1.5.30	registerWriteS32()	58
4.1.5.31	registerWriteU64()	59
4.1.5.32	registerWriteS64()	60
4.1.5.33	registerWriteF32()	60
4.1.5.34	registerWriteF64()	61

4.1.5.35	registerWriteAscii()	61
4.1.5.36	registerWriteRead()	62
4.1.5.37	registerWriteReadU8()	63
4.1.5.38	registerWriteReadS8()	63
4.1.5.39	registerWriteReadU16()	64
4.1.5.40	registerWriteReadS16()	65
4.1.5.41	registerWriteReadU32()	65
4.1.5.42	registerWriteReadS32()	66
4.1.5.43	registerWriteReadU64()	67
4.1.5.44	registerWriteReadS64()	67
4.1.5.45	registerWriteReadF32()	68
4.1.5.46	registerWriteReadF64()	69
4.1.5.47	registerWriteReadAscii()	69
4.1.5.48	deviceGetType()	70
4.1.5.49	deviceGetSysType()	71
4.1.5.50	deviceGetPartNumberStr()	71
4.1.5.51	deviceGetPCBVersion()	72
4.1.5.52	deviceGetStatusBits()	72
4.1.5.53	deviceGetErrorCode()	73
4.1.5.54	deviceGetBootloaderVersion()	73
4.1.5.55	deviceGetBootloaderVersionStr()	74
4.1.5.56	deviceGetFirmwareVersion()	74
4.1.5.57	deviceGetFirmwareVersionStr()	75
4.1.5.58	deviceGetModuleSerialNumberStr()	75
4.1.5.59	deviceGetPCBSerialNumberStr()	76
4.1.5.60	deviceCreate()	77
4.1.5.61	deviceExists()	77
4.1.5.62	deviceRemove()	77
4.1.5.63	deviceRemoveAll()	78
4.1.5.64	deviceGetAllTypes()	78

---

4.1.5.65	deviceGetMode()	79
4.1.5.66	deviceGetLive()	79
4.1.5.67	deviceSetLive()	80
4.1.5.68	registerCreate()	80
4.1.5.69	registerExists()	81
4.1.5.70	registerRemove()	81
4.1.5.71	registerRemoveAll()	82
4.1.5.72	registerGetAll()	82
4.1.5.73	setCallbackPtrPortInfo()	83
4.1.5.74	setCallbackPtrDeviceInfo()	83
4.1.5.75	setCallbackPtrRegisterInfo()	83
4.1.5.76	setLVUserEventPortInfo()	83
4.1.5.77	setLVUserEventDeviceInfo()	84
4.1.5.78	setLVUserEventRegisterInfo()	84
<b>Index</b>		<b>85</b>



# Chapter 1

## Data Structure Index

### 1.1 Data Structures

Here are the data structures with brief descriptions:

- [LvDeviceStatusStruct](#)  
LvDeviceStatusStruct, A LabView userevent data package . . . . . 5
- [LvPortStatusStruct](#)  
LvPortStatusStruct, A LabView userevent data package . . . . . 6
- [LvRegisterStatusStruct](#)  
LvRegisterStatusStruct, A LabView userevent data package . . . . . 7
- [tDateTimeStruct](#)  
The tDateTime struct 24 hour format . . . . . 9
- [tParamSetStruct](#)  
The tParameterSet struct . . . . . 10



## Chapter 2

# File Index

### 2.1 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 . . . [13](#)



## Chapter 3

# Data Structure Documentation

### 3.1 lvDeviceStatusStruct Struct Reference

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

#### 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.*

#### 3.1.1 Detailed Description

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

#### 3.1.2 Field Documentation

##### 3.1.2.1 portname

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

Zero terminated string giving the originating portname.

### 3.1.2.2 devId

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

The originating device id (module address).

### 3.1.2.3 status

```
DeviceStatusTypes lvDeviceStatusStruct::status
```

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

### 3.1.2.4 devDataLen

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

Number of databytes in devData.

### 3.1.2.5 devData

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

device data as specified in status.

## 3.2 lvPortStatusStruct Struct Reference

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

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

### 3.2.1 Detailed Description

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

## 3.2.2 Field Documentation

### 3.2.2.1 portname

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

Zero terminated string giving the originating portname.

### 3.2.2.2 status

```
PortStatusTypes lvPortStatusStruct::status
```

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

### 3.2.2.3 curScanAdr

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

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

### 3.2.2.4 maxScanAdr

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

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

### 3.2.2.5 foundType

```
unsigned char lvPortStatusStruct::foundType
```

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

## 3.3 lvRegisterStatusStruct Struct Reference

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

## 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.*

### 3.3.1 Detailed Description

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

### 3.3.2 Field Documentation

#### 3.3.2.1 portname

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

Zero terminated string giving the originating portname.

#### 3.3.2.2 devId

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

The originating device id (module address).

#### 3.3.2.3 regId

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

The originating register id.

#### 3.3.2.4 status

```
RegisterStatusTypes lvRegisterStatusStruct::status
```

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

### 3.3.2.5 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`.

### 3.3.2.6 regDataLen

`unsigned char` lvRegisterStatusStruct::regDataLen

Number of databytes.

### 3.3.2.7 regData

`unsigned char` lvRegisterStatusStruct::regData[255]

The register data.

## 3.4 tDateTimeStruct Struct Reference

The `tDateTime` struct 24 hour format.

### 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.*

### 3.4.1 Detailed Description

The `tDateTime` struct 24 hour format.

### 3.4.2 Field Documentation

#### 3.4.2.1 Sec

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

Seconds.

#### 3.4.2.2 Min

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

Minutes.

#### 3.4.2.3 Hour

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

Hours.

#### 3.4.2.4 Day

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

Day.

#### 3.4.2.5 Month

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

Months.

#### 3.4.2.6 Year

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

Years.

### 3.5 tParamSetStruct Struct Reference

The tParameterSet struct.

## 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.*

### 3.5.1 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

### 3.5.2 Field Documentation

#### 3.5.2.1 Unit

`ParamSetUnitTypes` `tParamSetStruct::Unit`

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

#### 3.5.2.2 ErrorHandler

`unsigned char` `tParamSetStruct::ErrorHandler`

Warning/Errorhandler not used.

### 3.5.2.3 StartVal

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

Setpoint for Settings parameterset, unused in Measurement parametersets.

### 3.5.2.4 FactoryVal

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

Factory Setpoint for Settings parameterset, unused in Measurement parametersets.

### 3.5.2.5 ULimit

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

Upper limit.

### 3.5.2.6 LLimit

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

Lower limit.

### 3.5.2.7 Numerator

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

Numerator(X) for calculation.

### 3.5.2.8 Denominator

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

Denominator(Y) for calculation.

### 3.5.2.9 Offset

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

Offset for calculation.

# Chapter 4

## File Documentation

### 4.1 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, RegResultCRCErr = 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, DeviceSystemTypeChanged = 11` }  
*The tDeviceStatusTypes enum.*
- enum `tRegisterStatusTypes` {  
`RegSuccess = 0, RegBusy = 1, RegNacked = 2, RegCRCErr = 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, Unitdnm = 15,`  
`Unitnm = 16, UnitPerCent = 17, UnitPerMille = 18, UnitmA = 19,`  
`UnitdmA = 20, UnitRPM = 21, UnitdBm = 22, UnitcBm = 23,`  
`UnitmBm = 24, UnitdB = 25, UnitcB = 26, UnitmB = 27,`  
`Unitdpm = 28, UnitcV = 29, UnitdV = 30, UnitIm = 31,`  
`Unitdlm = 32, Unitclm = 33, Unitmlm = 34 }`

*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 `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 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 \* [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 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 \* [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 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

- 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 char 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.*

### 4.1.1 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

**See also**

<http://www.nktpotonics.com/lasers-fibers/en/support/software-drivers/>

**4.1.2 Macro Definition Documentation****4.1.2.1 NKTPDLL\_EXPORT**

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

**4.1.3 Typedef Documentation****4.1.3.1 PortResultTypes**

```
typedef unsigned char PortResultTypes
```

**4.1.3.2 P2PPortResultTypes**

```
typedef unsigned char P2PPortResultTypes
```

**4.1.3.3 DeviceResultTypes**

```
typedef unsigned char DeviceResultTypes
```

**4.1.3.4 DeviceModeTypes**

```
typedef unsigned char DeviceModeTypes
```

**4.1.3.5 RegisterResultTypes**

```
typedef unsigned char RegisterResultTypes
```

**4.1.3.6 RegisterDataTypes**

```
typedef unsigned char RegisterDataTypes
```

**4.1.3.7 RegisterPriorityTypes**

```
typedef unsigned char RegisterPriorityTypes
```

#### 4.1.3.8 PortStatusTypes

```
typedef unsigned char PortStatusTypes
```

#### 4.1.3.9 DeviceStatusTypes

```
typedef unsigned char DeviceStatusTypes
```

#### 4.1.3.10 RegisterStatusTypes

```
typedef unsigned char RegisterStatusTypes
```

#### 4.1.3.11 DateTimeType

```
typedef struct tDateTimeStruct DateTimeType
```

The tDateTime struct 24 hour format.

#### 4.1.3.12 ParamSetUnitTypes

```
typedef unsigned char ParamSetUnitTypes
```

#### 4.1.3.13 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

#### 4.1.3.14 GetAllPortsFuncPtr

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

#### 4.1.3.15 GetOpenPortsFuncPtr

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

#### 4.1.3.16 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)
```

#### 4.1.3.17 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)
```

#### 4.1.3.18 PointToPointPortDelFuncPtr

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

#### 4.1.3.19 OpenPortsFuncPtr

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

#### 4.1.3.20 ClosePortsFuncPtr

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

#### 4.1.3.21 SetLegacyBusScanningFuncPtr

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

#### 4.1.3.22 GetLegacyBusScanningFuncPtr

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

#### 4.1.3.23 getPortStatusFuncPtr

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

#### 4.1.3.24 getPortErrorMsgFuncPtr

```
typedef PortResultTypes(__cdecl * getPortErrorMsgFuncPtr) (const char *portname, char *errorMessage, unsigned short *maxLen)
```

#### 4.1.3.25 RegisterReadFuncPtr

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

#### 4.1.3.26 RegisterReadU8FuncPtr

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

#### 4.1.3.27 RegisterReadS8FuncPtr

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

#### 4.1.3.28 RegisterReadU16FuncPtr

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

#### 4.1.3.29 RegisterReadS16FuncPtr

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

#### 4.1.3.30 RegisterReadU32FuncPtr

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

#### 4.1.3.31 RegisterReadS32FuncPtr

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

#### 4.1.3.32 RegisterReadU64FuncPtr

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

#### 4.1.3.33 RegisterReadS64FuncPtr

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

#### 4.1.3.34 RegisterReadF32FuncPtr

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

#### 4.1.3.35 RegisterReadF64FuncPtr

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

#### 4.1.3.36 RegisterReadAsciiFuncPtr

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

#### 4.1.3.37 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)
```

#### 4.1.3.38 RegisterWriteU8FuncPtr

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

#### 4.1.3.39 RegisterWriteS8FuncPtr

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

#### 4.1.3.40 RegisterWriteU16FuncPtr

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

#### 4.1.3.41 RegisterWriteS16FuncPtr

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

#### 4.1.3.42 RegisterWriteU32FuncPtr

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

#### 4.1.3.43 RegisterWriteS32FuncPtr

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

#### 4.1.3.44 RegisterWriteU64FuncPtr

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

#### 4.1.3.45 RegisterWriteS64FuncPtr

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

#### 4.1.3.46 RegisterWriteF32FuncPtr

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

#### 4.1.3.47 RegisterWriteF64FuncPtr

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

#### 4.1.3.48 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)
```

#### 4.1.3.49 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)
```

#### 4.1.3.50 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)
```

#### 4.1.3.51 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)
```

#### 4.1.3.52 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)
```

#### 4.1.3.53 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)
```

#### 4.1.3.54 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)
```

#### 4.1.3.55 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)
```

#### 4.1.3.56 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)
```

#### 4.1.3.57 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)
```

#### 4.1.3.58 RegisterWriteReadF32FuncPtr

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

#### 4.1.3.59 RegisterWriteReadF64FuncPtr

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

#### 4.1.3.60 RegisterWriteReadAsciiFuncPtr

```
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 *maxLength, const short index)
```

#### 4.1.3.61 DeviceGetTypeFuncPtr

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

#### 4.1.3.62 DeviceGetSysTypeFuncPtr

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

#### 4.1.3.63 DeviceGetPartNumberStrFuncPtr

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

#### 4.1.3.64 DeviceGetPCBVersionFuncPtr

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

#### 4.1.3.65 DeviceGetStatusBitsFuncPtr

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

#### 4.1.3.66 DeviceGetErrorCodeFuncPtr

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

#### 4.1.3.67 DeviceGetBootloaderVersionFuncPtr

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

#### 4.1.3.68 DeviceGetBootloaderVersionStrFuncPtr

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

#### 4.1.3.69 DeviceGetFirmwareVersionFuncPtr

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

#### 4.1.3.70 DeviceGetFirmwareVersionStrFuncPtr

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

#### 4.1.3.71 DeviceGetModuleSerialNumberStrFuncPtr

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

#### 4.1.3.72 DeviceGetPCBSerialNumberStrFuncPtr

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

#### 4.1.3.73 DeviceCreateFuncPtr

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

#### 4.1.3.74 DeviceExistsFuncPtr

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

#### 4.1.3.75 DeviceRemoveFuncPtr

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

#### 4.1.3.76 DeviceRemoveAllFuncPtr

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

#### 4.1.3.77 DeviceGetAllTypesFuncPtr

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

#### 4.1.3.78 DeviceGetModeFuncPtr

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

#### 4.1.3.79 DeviceGetLiveFuncPtr

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

#### 4.1.3.80 DeviceSetLiveFuncPtr

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

#### 4.1.3.81 RegisterCreateFuncPtr

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

#### 4.1.3.82 RegisterExistsFuncPtr

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

#### 4.1.3.83 RegisterRemoveFuncPtr

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

#### 4.1.3.84 RegisterRemoveAllFuncPtr

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

#### 4.1.3.85 RegisterGetAllFuncPtr

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

#### 4.1.3.86 PortStatusCallbackFuncPtr

```
typedef void(__cdecl * PortStatusCallbackFuncPtr) (const char *portname, const PortStatusTypes status, const unsigned char curScanAdr, const unsigned char maxScanAdr, const unsigned char 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 <a href="#">PortStatusTypes</a> with a <a href="#">tPortStatusTypes</a> value.
<i>curScanAdr</i>	When status is <a href="#">PortScanProgress</a> or <a href="#">PortScanDeviceFound</a> this indicates the current module address scanned or found.
<i>maxScanAdr</i>	When status is <a href="#">PortScanProgress</a> or <a href="#">PortScanDeviceFound</a> this indicates the last module address to be scanned.
<i>foundType</i>	When status is <a href="#">PortScanDeviceFound</a> this value will represent the found module type.

##### 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.

#### 4.1.3.87 SetCallbackPtrPortInfoFuncPtr

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

#### 4.1.3.88 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 <a href="#">DeviceStatusTypes</a> with a <a href="#">tDeviceStatusTypes</a> value.

## 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.

## 4.1.3.89 SetCallbackPtrDeviceInfoFuncPtr

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

## 4.1.3.90 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>status</i>	The current register status as a <a href="#">RegisterStatusTypes</a> with a <a href="#">tRegisterStatusTypes</a> value.
<i>regType</i>	The <a href="#">RegisterDataTypes</a> , 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.

## 4.1.3.91 SetCallbackPtrRegisterInfoFuncPtr

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

## 4.1.3.92 LabViewPortStatusType

```
typedef struct lvPortStatusStruct LabViewPortStatusType
```

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

#### 4.1.3.93 SetLVUserEventPortInfoFuncPtr

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

#### 4.1.3.94 LabViewDeviceStatusType

```
typedef struct lvDeviceStatusStruct LabViewDeviceStatusType
```

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

#### 4.1.3.95 SetLVUserEventDeviceInfoFuncPtr

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

#### 4.1.3.96 LabViewRegisterStatusType

```
typedef struct lvRegisterStatusStruct LabViewRegisterStatusType
```

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

#### 4.1.3.97 SetLVUserEventRegisterInfoFuncPtr

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

### 4.1.4 Enumeration Type Documentation

#### 4.1.4.1 tPortResultTypes

```
enum tPortResultTypes
```

The tPortResultTypes enum.

##### Enumerator

OPSuccess	0 - Successfull operation.
OPFailed	1 - The <a href="#">openPorts</a> 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.

#### 4.1.4.2 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.

## 4.1.4.3 tDeviceResultTypes

```
enum tDeviceResultTypes
```

The tDeviceResultTypes enum.

## Enumerator

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

## 4.1.4.4 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.
DevModeTimeout	6 - The connection to the device has been lost.
DevModeUpload	7 - The device is in upload mode and can not be used normally.

## 4.1.4.5 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 <a href="#">registerCreate</a> has failed.
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.
RegResultCRCErr	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 middel 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.

## 4.1.4.6 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).

## Enumerator

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. <a href="#">ParameterSetType</a>
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).
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. <a href="#">DateTimeType</a>

## 4.1.4.7 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.

## 4.1.4.8 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.

## 4.1.4.9 tDeviceStatusTypes

```
enum tDeviceStatusTypes
```

The tDeviceStatusTypes enum.

## Enumerator

DeviceModeChanged	0 - devData contains 1 unsigned byte <a href="#">DeviceModeTypes</a>
DeviceLiveChanged	1 - devData contains 1 unsigned byte, 0=live off, 1=live on.
DeviceTypeChanged	2 - devData contains 1 unsigned byte with DeviceType (module type).
DevicePartNumberChanged	3 - devData contains a zero terminated string with partnumber.
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).

## 4.1.4.10 tRegisterStatusTypes

```
enum tRegisterStatusTypes
```

The tRegisterStatusTypes enum.

## Enumerator

RegSuccess	0 - Register operation was successful.
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.

## 4.1.4.11 tParamSetUnitTypes

```
enum tParamSetUnitTypes
```

The tParamSetUnitTypes enum.

## Enumerator

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

## Enumerator

UnituA	3 - $\mu$ A
UnitmA	4 - mA
UnitA	5 - A
UnituW	6 - $\mu$ 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
Unitdm	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

## 4.1.5 Function Documentation

## 4.1.5.1 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.

## 4.1.5.2 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.

## 4.1.5.3 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.
<i>protocol</i>	<ul style="list-style-type: none"> <li>• 0 Specifies TCP protocol.</li> <li>• 1 Specifies UDP protocol.</li> </ul>
<i>msTimeout</i>	Telegram timeout value in milliseconds, typically set to 100ms.

## Returns

[tP2PPortResultTypes](#)

## 4.1.5.4 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 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 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"> <li>• 0 Specifies TCP protocol.</li> <li>• 1 Specifies UDP protocol.</li> </ul>
<i>msTimeout</i>	Pointer to a preallocated char where the function will store the timeout value.

#### Returns

[tP2PPortResultTypes](#)

#### 4.1.5.5 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](#)

## 4.1.5.6 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"> <li>• 0 the openPorts function only opens the port. Busscanning and device creation is NOT automatically handled.</li> <li>• 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.</li> </ul>
<i>liveMode</i>	<ul style="list-style-type: none"> <li>• 0 the openPorts function disables the continuously monitoring of the registers. No callback possible on register changes. Use <a href="#">registerRead</a>, <a href="#">registerWrite</a> &amp; <a href="#">registerWriteRead</a> functions.</li> <li>• 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.</li> </ul>

## 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.

## 4.1.5.7 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](#).

**4.1.5.8 setLegacyBusScanning()**

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

Sets legacy busscanning on or off.

**Parameters**

<i>legacyScanning</i>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• 1 the busscanning is set to legacy mode and fixes the masterId at address 66(0x42). Some older modules does not accept masterIds other than 66(0x42).</li> </ul>
-----------------------	--

**See also**

[getLegacyBusScanning](#)

**4.1.5.9 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](#)

**4.1.5.10 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 <a href="#">PortStatusTypes</a> where the function will store the port status.

## Returns

[tPortResultTypes](#)

4.1.5.11 `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](#)

4.1.5.12 `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.

## Parameters

<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.

## 4.1.5.13 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.

## 4.1.5.14 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.

## 4.1.5.15 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.

**4.1.5.16 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.

**4.1.5.17 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.

## 4.1.5.18 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.

## 4.1.5.19 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.

## 4.1.5.20 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.

**4.1.5.21 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.

**4.1.5.22 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.

**4.1.5.23 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.

## 4.1.5.24 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 $\geq 0$ 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.

## 4.1.5.25 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).
-----------------	--

## Parameters

<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 $\geq 0$ 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.

## 4.1.5.26 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 $\geq 0$ 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.

## 4.1.5.27 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 $\geq 0$ 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.

## 4.1.5.28 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 $\geq 0$ 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.

**4.1.5.29 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 $\geq 0$ 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.

**4.1.5.30 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 $\geq 0$ 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.

## 4.1.5.31 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 $\geq 0$ 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.

#### 4.1.5.32 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 $\geq 0$ 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.

#### 4.1.5.33 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 $\geq 0$ 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.

**4.1.5.34 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 $\geq 0$ 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.

**4.1.5.35 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"> <li>• 0 Do NOT append End Of Line character (a null character) to the string.</li> <li>• 1 Append End Of Line character to the string.</li> </ul>
<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 $\geq 0$ 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.

## 4.1.5.36 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 $\geq 0$ 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.

**4.1.5.37 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 $\geq 0$ 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.

**4.1.5.38 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 $\geq 0$ 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.

#### 4.1.5.39 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 $\geq 0$ 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.

**4.1.5.40 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 $\geq 0$ 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.

**4.1.5.41 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 $\geq 0$ 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.

## 4.1.5.42 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 $\geq 0$ 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.

**4.1.5.43 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 $\geq 0$ 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.

**4.1.5.44 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 $\geq 0$ 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.

## 4.1.5.45 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 $\geq 0$ 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.

**4.1.5.46 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 $\geq 0$ 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.

**4.1.5.47 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"> <li>• 0 Do NOT append End Of Line character (a null character) to the string.</li> <li>• 1 Append End Of Line character to the string.</li> </ul>
<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 $\geq 0$ 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.

4.1.5.48 `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.

**4.1.5.49 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.

**4.1.5.50 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.

**4.1.5.51 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.

**4.1.5.52 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.

**4.1.5.53 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.

**4.1.5.54 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.

**4.1.5.55 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.

**4.1.5.56 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.

**4.1.5.57 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.

**4.1.5.58 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.

**4.1.5.59 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.

## 4.1.5.60 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 immediatedly 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"> <li>• 0 Don't wait for the device being ready.</li> <li>• 1 Wait up to 2 seconds for the device to complete its analyze cycle (All standard registers being successfully read).</li> </ul>

## Returns

A status result value [tDeviceResultTypes](#)

## 4.1.5.61 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>	Pointer to an unsigned char where the function will store the exists status. <ul style="list-style-type: none"> <li>• 0 Device does not exists.</li> <li>• 1 Device exists.</li> </ul>

## Returns

A status result value [tDeviceResultTypes](#)

## 4.1.5.62 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](#)

#### 4.1.5.63 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](#)

#### 4.1.5.64 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](#)

**4.1.5.65 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 <a href="#">DeviceModeTypes</a> where the function stores the device mode value <a href="#">tDeviceModeTypes</a>

**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.

**4.1.5.66 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"> <li>• 0 liveMode off</li> <li>• 1 liveMode on</li> </ul>

**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.

**4.1.5.67 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"> <li>• 0 liveMode off</li> <li>• 1 liveMode on</li> </ul>

**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.

**4.1.5.68 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 immediately 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 <a href="#">tRegisterPriorityTypes</a> (monitoring priority).
<i>dataType</i>	The <a href="#">tRegisterDataTypes</a> , not used internally but could be used in a common callback function to determine data type.

## Returns

A status result value [tRegisterResultTypes](#)

## 4.1.5.69 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"> <li>• 0 Register does not exists.</li> <li>• 1 Register exists.</li> </ul>

## Returns

A status result value [tRegisterResultTypes](#)

## 4.1.5.70 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](#)

## 4.1.5.71 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](#)

## 4.1.5.72 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](#)

**4.1.5.73 setCallbackPtrPortInfo()**

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

Enables/Disables callback for port status changes.

**Parameters**

<i>callback</i>	The <a href="#">PortStatusCallbackFuncPtr</a> function pointer. Disable callbacks by parsing in a zero value.
-----------------	---

**4.1.5.74 setCallbackPtrDeviceInfo()**

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

Enables/Disables callback for device status changes.

**Parameters**

<i>callback</i>	The <a href="#">DeviceStatusCallbackFuncPtr</a> function pointer. Disable callbacks by parsing in a zero value.
-----------------	---

**4.1.5.75 setCallbackPtrRegisterInfo()**

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

Enables/Disables callback for register status changes.

**Parameters**

<i>callback</i>	The <a href="#">RegisterStatusCallbackFuncPtr</a> function pointer. Disable callbacks by parsing in a zero value.
-----------------	---

**4.1.5.76 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 usevent type.
-----------------------	---

#### 4.1.5.77 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.
-----------------------	---

#### 4.1.5.78 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.
-----------------------	---

# Index

- closePorts
  - NKTPDLL.h, [45](#)
- ClosePortsFuncPtr
  - NKTPDLL.h, [26](#)
- curScanAdr
  - lvPortStatusStruct, [7](#)
  
- DateTimeType
  - NKTPDLL.h, [25](#)
- Day
  - tDateTimeStruct, [10](#)
- Denominator
  - tParamSetStruct, [12](#)
- devData
  - lvDeviceStatusStruct, [6](#)
- devDataLen
  - lvDeviceStatusStruct, [6](#)
- devId
  - lvDeviceStatusStruct, [5](#)
  - lvRegisterStatusStruct, [8](#)
- deviceCreate
  - NKTPDLL.h, [76](#)
- DeviceCreateFuncPtr
  - NKTPDLL.h, [32](#)
- deviceExists
  - NKTPDLL.h, [77](#)
- DeviceExistsFuncPtr
  - NKTPDLL.h, [32](#)
- deviceGetAllTypes
  - NKTPDLL.h, [78](#)
- DeviceGetAllTypesFuncPtr
  - NKTPDLL.h, [33](#)
- deviceGetBootloaderVersion
  - NKTPDLL.h, [73](#)
- DeviceGetBootloaderVersionFuncPtr
  - NKTPDLL.h, [32](#)
- deviceGetBootloaderVersionStr
  - NKTPDLL.h, [74](#)
- DeviceGetBootloaderVersionStrFuncPtr
  - NKTPDLL.h, [32](#)
- deviceGetErrorCode
  - NKTPDLL.h, [73](#)
- DeviceGetErrorCodeFuncPtr
  - NKTPDLL.h, [31](#)
- deviceGetFirmwareVersion
  - NKTPDLL.h, [74](#)
- DeviceGetFirmwareVersionFuncPtr
  - NKTPDLL.h, [32](#)
- deviceGetFirmwareVersionStr
  - NKTPDLL.h, [75](#)
  
- DeviceGetFirmwareVersionStrFuncPtr
  - NKTPDLL.h, [32](#)
- deviceGetLive
  - NKTPDLL.h, [79](#)
- DeviceGetLiveFuncPtr
  - NKTPDLL.h, [33](#)
- deviceGetMode
  - NKTPDLL.h, [79](#)
- DeviceGetModeFuncPtr
  - NKTPDLL.h, [33](#)
- deviceGetModuleSerialNumberStr
  - NKTPDLL.h, [75](#)
- DeviceGetModuleSerialNumberStrFuncPtr
  - NKTPDLL.h, [32](#)
- deviceGetPCBSerialNumberStr
  - NKTPDLL.h, [76](#)
- DeviceGetPCBSerialNumberStrFuncPtr
  - NKTPDLL.h, [32](#)
- deviceGetPCBVersion
  - NKTPDLL.h, [72](#)
- DeviceGetPCBVersionFuncPtr
  - NKTPDLL.h, [31](#)
- deviceGetPartNumberStr
  - NKTPDLL.h, [71](#)
- DeviceGetPartNumberStrFuncPtr
  - NKTPDLL.h, [31](#)
- deviceGetStatusBits
  - NKTPDLL.h, [72](#)
- DeviceGetStatusBitsFuncPtr
  - NKTPDLL.h, [31](#)
- deviceGetSysType
  - NKTPDLL.h, [71](#)
- DeviceGetSysTypeFuncPtr
  - NKTPDLL.h, [31](#)
- deviceGetType
  - NKTPDLL.h, [70](#)
- DeviceGetTypeFuncPtr
  - NKTPDLL.h, [31](#)
- DeviceModeTypes
  - NKTPDLL.h, [24](#)
- deviceRemove
  - NKTPDLL.h, [77](#)
- deviceRemoveAll
  - NKTPDLL.h, [78](#)
- DeviceRemoveAllFuncPtr
  - NKTPDLL.h, [33](#)
- DeviceRemoveFuncPtr
  - NKTPDLL.h, [32](#)
- DeviceResultTypes

- NKTPDLL.h, 24
- deviceSetLive
  - NKTPDLL.h, 80
- DeviceSetLiveFuncPtr
  - NKTPDLL.h, 33
- DeviceStatusCallbackFuncPtr
  - NKTPDLL.h, 34
- DeviceStatusTypes
  - NKTPDLL.h, 25
- ErrorHandler
  - tParamSetStruct, 11
- FactoryVal
  - tParamSetStruct, 12
- foundType
  - lvPortStatusStruct, 7
- getAllPorts
  - NKTPDLL.h, 42
- GetAllPortsFuncPtr
  - NKTPDLL.h, 25
- getLegacyBusScanning
  - NKTPDLL.h, 46
- GetLegacyBusScanningFuncPtr
  - NKTPDLL.h, 26
- getOpenPorts
  - NKTPDLL.h, 42
- GetOpenPortsFuncPtr
  - NKTPDLL.h, 25
- getPortErrorMsg
  - NKTPDLL.h, 47
- getPortErrorMsgFuncPtr
  - NKTPDLL.h, 26
- getPortStatus
  - NKTPDLL.h, 46
- getPortStatusFuncPtr
  - NKTPDLL.h, 26
- Hour
  - tDateTimeStruct, 10
- LLimit
  - tParamSetStruct, 12
- LabViewDeviceStatusType
  - NKTPDLL.h, 36
- LabViewPortStatusType
  - NKTPDLL.h, 35
- LabViewRegisterStatusType
  - NKTPDLL.h, 36
- lvDeviceStatusStruct, 5
  - devData, 6
  - devDataLen, 6
  - devId, 5
  - portname, 5
  - status, 6
- lvPortStatusStruct, 6
  - curScanAdr, 7
  - foundType, 7
  - maxScanAdr, 7
  - portname, 7
  - status, 7
- lvRegisterStatusStruct, 7
  - devId, 8
  - portname, 8
  - regData, 9
  - regDataLen, 9
  - regId, 8
  - regType, 8
  - status, 8
- maxScanAdr
  - lvPortStatusStruct, 7
- Min
  - tDateTimeStruct, 10
- Month
  - tDateTimeStruct, 10
- NKTPDLL.h, 13
  - closePorts, 45
  - ClosePortsFuncPtr, 26
  - DateTimeType, 25
  - deviceCreate, 76
  - DeviceCreateFuncPtr, 32
  - deviceExists, 77
  - DeviceExistsFuncPtr, 32
  - deviceGetAllTypes, 78
  - DeviceGetAllTypesFuncPtr, 33
  - deviceGetBootloaderVersion, 73
  - DeviceGetBootloaderVersionFuncPtr, 32
  - deviceGetBootloaderVersionStr, 74
  - DeviceGetBootloaderVersionStrFuncPtr, 32
  - deviceGetErrorCode, 73
  - DeviceGetErrorCodeFuncPtr, 31
  - deviceGetFirmwareVersion, 74
  - DeviceGetFirmwareVersionFuncPtr, 32
  - deviceGetFirmwareVersionStr, 75
  - DeviceGetFirmwareVersionStrFuncPtr, 32
  - deviceGetLive, 79
  - DeviceGetLiveFuncPtr, 33
  - deviceGetMode, 79
  - DeviceGetModeFuncPtr, 33
  - deviceGetModuleSerialNumberStr, 75
  - DeviceGetModuleSerialNumberStrFuncPtr, 32
  - deviceGetPCBSerialNumberStr, 76
  - DeviceGetPCBSerialNumberStrFuncPtr, 32
  - deviceGetPCBVersion, 72
  - DeviceGetPCBVersionFuncPtr, 31
  - deviceGetPartNumberStr, 71
  - DeviceGetPartNumberStrFuncPtr, 31
  - deviceGetStatusBits, 72
  - DeviceGetStatusBitsFuncPtr, 31
  - deviceGetSysType, 71
  - DeviceGetSysTypeFuncPtr, 31
  - deviceGetType, 70
  - DeviceGetTypeFuncPtr, 31
  - DeviceModeTypes, 24
  - deviceRemove, 77

deviceRemoveAll, 78  
DeviceRemoveAllFuncPtr, 33  
DeviceRemoveFuncPtr, 32  
DeviceResultTypes, 24  
deviceSetLive, 80  
DeviceSetLiveFuncPtr, 33  
DeviceStatusCallbackFuncPtr, 34  
DeviceStatusTypes, 25  
getAllPorts, 42  
GetAllPortsFuncPtr, 25  
getLegacyBusScanning, 46  
GetLegacyBusScanningFuncPtr, 26  
getOpenPorts, 42  
GetOpenPortsFuncPtr, 25  
getPortErrorMsg, 47  
getPortErrorMsgFuncPtr, 26  
getPortStatus, 46  
getPortStatusFuncPtr, 26  
LabViewDeviceStatusType, 36  
LabViewPortStatusType, 35  
LabViewRegisterStatusType, 36  
NKTPDLL\_EXPORT, 24  
openPorts, 44  
OpenPortsFuncPtr, 26  
P2PPortResultTypes, 24  
ParamSetUnitTypes, 25  
ParameterSetType, 25  
pointToPointPortAdd, 43  
PointToPointPortAddFuncPtr, 25  
pointToPointPortDel, 44  
PointToPointPortDelFuncPtr, 26  
pointToPointPortGet, 43  
PointToPointPortGetFuncPtr, 26  
PortResultTypes, 24  
PortStatusCallbackFuncPtr, 34  
PortStatusTypes, 24  
registerCreate, 80  
RegisterCreateFuncPtr, 33  
RegisterDataTypes, 24  
registerExists, 81  
RegisterExistsFuncPtr, 33  
registerGetAll, 82  
RegisterGetAllFuncPtr, 34  
RegisterPriorityTypes, 24  
registerRead, 47  
registerReadAscii, 54  
RegisterReadAsciiFuncPtr, 28  
registerReadF32, 53  
RegisterReadF32FuncPtr, 27  
registerReadF64, 53  
RegisterReadF64FuncPtr, 28  
RegisterReadFuncPtr, 26  
registerReadS16, 50  
RegisterReadS16FuncPtr, 27  
registerReadS32, 51  
RegisterReadS32FuncPtr, 27  
registerReadS64, 52  
RegisterReadS64FuncPtr, 27  
registerReadS8, 48  
RegisterReadS8FuncPtr, 27  
registerReadU16, 49  
RegisterReadU16FuncPtr, 27  
registerReadU32, 50  
RegisterReadU32FuncPtr, 27  
registerReadU64, 51  
RegisterReadU64FuncPtr, 27  
registerReadU8, 48  
RegisterReadU8FuncPtr, 27  
registerRemove, 81  
registerRemoveAll, 82  
RegisterRemoveAllFuncPtr, 33  
RegisterRemoveFuncPtr, 33  
RegisterResultTypes, 24  
RegisterStatusCallbackFuncPtr, 35  
RegisterStatusTypes, 25  
registerWrite, 54  
registerWriteAscii, 61  
RegisterWriteAsciiFuncPtr, 29  
registerWriteF32, 60  
RegisterWriteF32FuncPtr, 29  
registerWriteF64, 61  
RegisterWriteF64FuncPtr, 29  
RegisterWriteFuncPtr, 28  
registerWriteRead, 62  
registerWriteReadAscii, 69  
RegisterWriteReadAsciiFuncPtr, 31  
registerWriteReadF32, 68  
RegisterWriteReadF32FuncPtr, 30  
registerWriteReadF64, 69  
RegisterWriteReadF64FuncPtr, 31  
RegisterWriteReadFuncPtr, 29  
registerWriteReadS16, 65  
RegisterWriteReadS16FuncPtr, 30  
registerWriteReadS32, 66  
RegisterWriteReadS32FuncPtr, 30  
registerWriteReadS64, 67  
RegisterWriteReadS64FuncPtr, 30  
registerWriteReadS8, 63  
RegisterWriteReadS8FuncPtr, 30  
registerWriteReadU16, 64  
RegisterWriteReadU16FuncPtr, 30  
registerWriteReadU32, 65  
RegisterWriteReadU32FuncPtr, 30  
registerWriteReadU64, 67  
RegisterWriteReadU64FuncPtr, 30  
registerWriteReadU8, 63  
RegisterWriteReadU8FuncPtr, 29  
registerWriteS16, 57  
RegisterWriteS16FuncPtr, 28  
registerWriteS32, 58  
RegisterWriteS32FuncPtr, 29  
registerWriteS64, 59  
RegisterWriteS64FuncPtr, 29  
registerWriteS8, 56  
RegisterWriteS8FuncPtr, 28  
registerWriteU16, 56

- RegisterWriteU16FuncPtr, 28
- registerWriteU32, 58
- RegisterWriteU32FuncPtr, 28
- registerWriteU64, 59
- RegisterWriteU64FuncPtr, 29
- registerWriteU8, 55
- RegisterWriteU8FuncPtr, 28
- setCallbackPtrDeviceInfo, 83
- SetCallbackPtrDeviceInfoFuncPtr, 35
- setCallbackPtrPortInfo, 83
- SetCallbackPtrPortInfoFuncPtr, 34
- setCallbackPtrRegisterInfo, 83
- SetCallbackPtrRegisterInfoFuncPtr, 35
- setLVUserEventDeviceInfo, 84
- SetLVUserEventDeviceInfoFuncPtr, 36
- setLVUserEventPortInfo, 83
- SetLVUserEventPortInfoFuncPtr, 35
- setLVUserEventRegisterInfo, 84
- SetLVUserEventRegisterInfoFuncPtr, 36
- setLegacyBusScanning, 46
- SetLegacyBusScanningFuncPtr, 26
- tDeviceModeTypes, 38
- tDeviceResultTypes, 38
- tDeviceStatusTypes, 41
- tP2PPortResultTypes, 36
- tParamSetUnitTypes, 41
- tPortResultTypes, 36
- tPortStatusTypes, 40
- tRegisterDataTypes, 39
- tRegisterPriorityTypes, 40
- tRegisterResultTypes, 38
- tRegisterStatusTypes, 41
- NKTPDLL\_EXPORT
  - NKTPDLL.h, 24
- Numerator
  - tParamSetStruct, 12
- Offset
  - tParamSetStruct, 12
- openPorts
  - NKTPDLL.h, 44
- OpenPortsFuncPtr
  - NKTPDLL.h, 26
- P2PPortResultTypes
  - NKTPDLL.h, 24
- ParamSetUnitTypes
  - NKTPDLL.h, 25
- ParameterSetType
  - NKTPDLL.h, 25
- pointToPointPortAdd
  - NKTPDLL.h, 43
- PointToPointPortAddFuncPtr
  - NKTPDLL.h, 25
- pointToPointPortDel
  - NKTPDLL.h, 44
- PointToPointPortDelFuncPtr
  - NKTPDLL.h, 26
- pointToPointPortGet
  - NKTPDLL.h, 43
- PointToPointPortGetFuncPtr
  - NKTPDLL.h, 26
- PortResultTypes
  - NKTPDLL.h, 24
- PortStatusCallbackFuncPtr
  - NKTPDLL.h, 34
- PortStatusTypes
  - NKTPDLL.h, 24
- portname
  - lvDeviceStatusStruct, 5
  - lvPortStatusStruct, 7
  - lvRegisterStatusStruct, 8
- regData
  - lvRegisterStatusStruct, 9
- regDataLen
  - lvRegisterStatusStruct, 9
- regId
  - lvRegisterStatusStruct, 8
- regType
  - lvRegisterStatusStruct, 8
- registerCreate
  - NKTPDLL.h, 80
- RegisterCreateFuncPtr
  - NKTPDLL.h, 33
- RegisterDataTypes
  - NKTPDLL.h, 24
- registerExists
  - NKTPDLL.h, 81
- RegisterExistsFuncPtr
  - NKTPDLL.h, 33
- registerGetAll
  - NKTPDLL.h, 82
- RegisterGetAllFuncPtr
  - NKTPDLL.h, 34
- RegisterPriorityTypes
  - NKTPDLL.h, 24
- registerRead
  - NKTPDLL.h, 47
- registerReadAscii
  - NKTPDLL.h, 54
- RegisterReadAsciiFuncPtr
  - NKTPDLL.h, 28
- registerReadF32
  - NKTPDLL.h, 53
- RegisterReadF32FuncPtr
  - NKTPDLL.h, 27
- registerReadF64
  - NKTPDLL.h, 53
- RegisterReadF64FuncPtr
  - NKTPDLL.h, 28
- RegisterReadFuncPtr
  - NKTPDLL.h, 26
- registerReadS16
  - NKTPDLL.h, 50
- RegisterReadS16FuncPtr
  - NKTPDLL.h, 27
- registerReadS32

NKTPDLL.h, [51](#)  
RegisterReadS32FuncPtr  
NKTPDLL.h, [27](#)  
registerReadS64  
NKTPDLL.h, [52](#)  
RegisterReadS64FuncPtr  
NKTPDLL.h, [27](#)  
registerReadS8  
NKTPDLL.h, [48](#)  
RegisterReadS8FuncPtr  
NKTPDLL.h, [27](#)  
registerReadU16  
NKTPDLL.h, [49](#)  
RegisterReadU16FuncPtr  
NKTPDLL.h, [27](#)  
registerReadU32  
NKTPDLL.h, [50](#)  
RegisterReadU32FuncPtr  
NKTPDLL.h, [27](#)  
registerReadU64  
NKTPDLL.h, [51](#)  
RegisterReadU64FuncPtr  
NKTPDLL.h, [27](#)  
registerReadU8  
NKTPDLL.h, [48](#)  
RegisterReadU8FuncPtr  
NKTPDLL.h, [27](#)  
registerRemove  
NKTPDLL.h, [81](#)  
registerRemoveAll  
NKTPDLL.h, [82](#)  
RegisterRemoveAllFuncPtr  
NKTPDLL.h, [33](#)  
RegisterRemoveFuncPtr  
NKTPDLL.h, [33](#)  
RegisterResultTypes  
NKTPDLL.h, [24](#)  
RegisterStatusCallbackFuncPtr  
NKTPDLL.h, [35](#)  
RegisterStatusTypes  
NKTPDLL.h, [25](#)  
registerWrite  
NKTPDLL.h, [54](#)  
registerWriteAscii  
NKTPDLL.h, [61](#)  
RegisterWriteAsciiFuncPtr  
NKTPDLL.h, [29](#)  
registerWriteF32  
NKTPDLL.h, [60](#)  
RegisterWriteF32FuncPtr  
NKTPDLL.h, [29](#)  
registerWriteF64  
NKTPDLL.h, [61](#)  
RegisterWriteF64FuncPtr  
NKTPDLL.h, [29](#)  
RegisterWriteFuncPtr  
NKTPDLL.h, [28](#)  
registerWriteRead  
NKTPDLL.h, [62](#)  
registerWriteReadAscii  
NKTPDLL.h, [69](#)  
RegisterWriteReadAsciiFuncPtr  
NKTPDLL.h, [31](#)  
registerWriteReadF32  
NKTPDLL.h, [68](#)  
RegisterWriteReadF32FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadF64  
NKTPDLL.h, [69](#)  
RegisterWriteReadF64FuncPtr  
NKTPDLL.h, [31](#)  
RegisterWriteReadFuncPtr  
NKTPDLL.h, [29](#)  
registerWriteReadS16  
NKTPDLL.h, [65](#)  
RegisterWriteReadS16FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadS32  
NKTPDLL.h, [66](#)  
RegisterWriteReadS32FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadS64  
NKTPDLL.h, [67](#)  
RegisterWriteReadS64FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadS8  
NKTPDLL.h, [63](#)  
RegisterWriteReadS8FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadU16  
NKTPDLL.h, [64](#)  
RegisterWriteReadU16FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadU32  
NKTPDLL.h, [65](#)  
RegisterWriteReadU32FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadU64  
NKTPDLL.h, [67](#)  
RegisterWriteReadU64FuncPtr  
NKTPDLL.h, [30](#)  
registerWriteReadU8  
NKTPDLL.h, [63](#)  
RegisterWriteReadU8FuncPtr  
NKTPDLL.h, [29](#)  
registerWriteS16  
NKTPDLL.h, [57](#)  
RegisterWriteS16FuncPtr  
NKTPDLL.h, [28](#)  
registerWriteS32  
NKTPDLL.h, [58](#)  
RegisterWriteS32FuncPtr  
NKTPDLL.h, [29](#)  
registerWriteS64  
NKTPDLL.h, [59](#)  
RegisterWriteS64FuncPtr

- NKTPDLL.h, 29
- registerWriteS8
  - NKTPDLL.h, 56
- RegisterWriteS8FuncPtr
  - NKTPDLL.h, 28
- registerWriteU16
  - NKTPDLL.h, 56
- RegisterWriteU16FuncPtr
  - NKTPDLL.h, 28
- registerWriteU32
  - NKTPDLL.h, 58
- RegisterWriteU32FuncPtr
  - NKTPDLL.h, 28
- registerWriteU64
  - NKTPDLL.h, 59
- RegisterWriteU64FuncPtr
  - NKTPDLL.h, 29
- registerWriteU8
  - NKTPDLL.h, 55
- RegisterWriteU8FuncPtr
  - NKTPDLL.h, 28
- Sec
  - tDateTimeStruct, 10
- setCallbackPtrDeviceInfo
  - NKTPDLL.h, 83
- SetCallbackPtrDeviceInfoFuncPtr
  - NKTPDLL.h, 35
- setCallbackPtrPortInfo
  - NKTPDLL.h, 83
- SetCallbackPtrPortInfoFuncPtr
  - NKTPDLL.h, 34
- setCallbackPtrRegisterInfo
  - NKTPDLL.h, 83
- SetCallbackPtrRegisterInfoFuncPtr
  - NKTPDLL.h, 35
- setLVUserEventDeviceInfo
  - NKTPDLL.h, 84
- SetLVUserEventDeviceInfoFuncPtr
  - NKTPDLL.h, 36
- setLVUserEventPortInfo
  - NKTPDLL.h, 83
- SetLVUserEventPortInfoFuncPtr
  - NKTPDLL.h, 35
- setLVUserEventRegisterInfo
  - NKTPDLL.h, 84
- SetLVUserEventRegisterInfoFuncPtr
  - NKTPDLL.h, 36
- setLegacyBusScanning
  - NKTPDLL.h, 46
- SetLegacyBusScanningFuncPtr
  - NKTPDLL.h, 26
- StartVal
  - tParamSetStruct, 11
- status
  - lvDeviceStatusStruct, 6
  - lvPortStatusStruct, 7
  - lvRegisterStatusStruct, 8
- tDateTimeStruct, 9
  - Day, 10
  - Hour, 10
  - Min, 10
  - Month, 10
  - Sec, 10
  - Year, 10
- tDeviceModeTypes
  - NKTPDLL.h, 38
- tDeviceResultTypes
  - NKTPDLL.h, 38
- tDeviceStatusTypes
  - NKTPDLL.h, 41
- tP2PPortResultTypes
  - NKTPDLL.h, 36
- tParamSetStruct, 10
  - Denominator, 12
  - ErrorHandler, 11
  - FactoryVal, 12
  - LLimit, 12
  - Numerator, 12
  - Offset, 12
  - StartVal, 11
  - ULimit, 12
  - Unit, 11
- tParamSetUnitTypes
  - NKTPDLL.h, 41
- tPortResultTypes
  - NKTPDLL.h, 36
- tPortStatusTypes
  - NKTPDLL.h, 40
- tRegisterDataTypes
  - NKTPDLL.h, 39
- tRegisterPriorityTypes
  - NKTPDLL.h, 40
- tRegisterResultTypes
  - NKTPDLL.h, 38
- tRegisterStatusTypes
  - NKTPDLL.h, 41
- ULimit
  - tParamSetStruct, 12
- Unit
  - tParamSetStruct, 11
- Year
  - tDateTimeStruct, 10