

SDK

Software Development Kit for NKT Photonics Instruments

Instruction Manual



Version: 2.1.3
Published: Oct 2018
Author: HCH, TOO, ETH, MKU
Copyright © 2018 by NKT Photonics A/S. All rights reserved. Reproduction or translations of any part of this work is prohibited.

Table of Contents

1	Introduction.....	5
2	Interbus Protocol Description.....	7
2.1	Physical	7
2.2	Telegram	11
2.3	Special character conversion.....	15
2.4	Communication Examples	16
2.5	Code Example, Transmission.....	18
2.6	Code Example, Reception	19
3	NKTPDLL	21
4	Examples.....	22
4.1	DLL_Example_CS & DLL_Example_VB.....	23
4.2	DLL_Example_CS_Callback & DLL_Example_VB_Callback	24
4.3	DLL_Example_Python.....	25
4.4	IB_Example_CPP(CLI) & IB_Example_CS	26
4.5	IB_Example_Qt (Using Qt framework).....	27
5	LabVIEW Driver API	28
5.1	Adding the NKTP LabVIEW VIs to the LabVIEW palettes.	28
5.2	NKTP LabVIEW API structure	28
6	Register Files.....	30
6.1	General Registers.....	32
6.2	Koheras AdjustiK/BoostiK System (K81-1 to K83-1)	33
6.3	Koheras ADJUSTIK/ACOUSTIK System (K822 / K852).....	34
6.3.1	General settings.....	34
6.3.2	Readouts	34
6.3.3	Modulation, master module.....	35
6.3.4	Modulation, laser modules	36
6.3.5	Ethernet	36
6.3.6	Special options.....	36
6.4	Koheras Basik Module (K80-1)	37
6.5	Koheras BASIK MIKRO Module (K0x2).....	39
6.5.1	General settings.....	39
6.5.2	Readouts	40
6.6	Koheras BASIK Module (K1x2)	41
6.6.1	General settings.....	41
6.6.2	Readouts	42
6.6.3	Modulation	43
6.7	BoostiK OEM Amplifier (N83)	44
6.8	SuperK EXTREME System (S4x2)	45
6.8.1	Main module	45
6.8.2	Front panel.....	46
6.8.3	Booster	47
6.9	RF Driver (A901) for SuperK SELECT.....	48
6.10	SuperK SELECT (A203)	50
6.11	SuperK VARIA (A301)	51
6.12	Extend UV (A351).....	52
6.13	SuperK COMPACT (S024)	53
6.14	aeroPULSE (P000).....	56
6.14.1	Control Unit.....	56
6.14.2	Amplifier Module	57
6.15	SuperK EVO (S2x1).....	59
7	Silabs USB Driver	61
8	Generic User Interface	63
8.1	Installing the software	63
8.2	Register File Path	66
8.3	Using the Generic User Interface Software	66
8.3.1	Front Panel	66
8.3.2	Starting Up.....	68
8.3.3	Controls	69
8.3.4	Readings.....	70
8.3.5	Error and Status.....	71
8.3.6	Graph.....	72
8.3.7	Functions	73
8.3.8	Loops	74
9	Appendix.....	76
9.1	LabVIEW Driver (Legacy)	76

9.1.1	Inputs and Outputs.....	76
9.1.2	PortParameters.....	78
9.1.3	InParameters	79
9.1.4	OutParameters.....	80
9.1.5	Using the NGSerialPort.....	82
9.1.6	Programming Examples.....	85
10	87

1 Introduction

Introduction

Please take the necessary time to read this software manual, which contains important information about how to get started with the Software Development Kit.

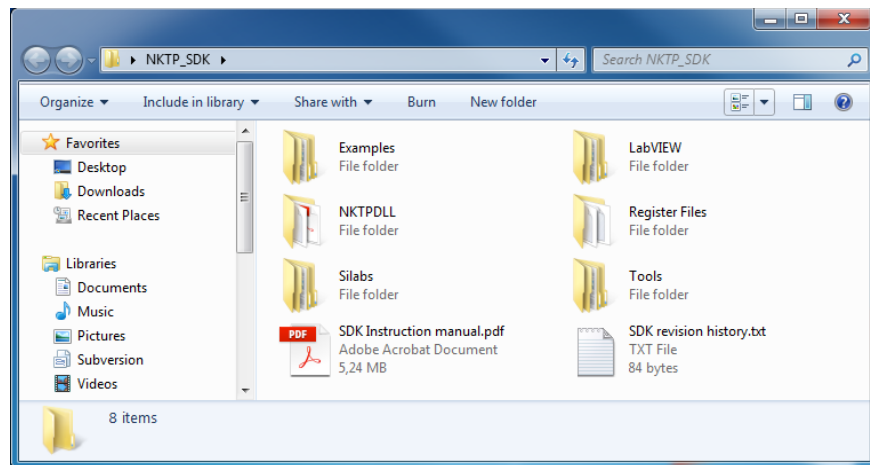
Warning

Laser products are as such potentially dangerous. Read the instruction manual for the laser, amplifier and/or accessory before continuing.



Contents

The Software Development Kit features this manual, a windows DLL, C++ source code, C# source code, a LabVIEW driver, register lists, USB driver and a generic user interface.



Manual

This manual covers a description of the NKT Photonics Interbus protocol, description of how to use the NKTPDLL and LabVIEW driver, how to read the register lists, how to install USB driver and how to install and use the Generic User Interface.

Protocol Description

The [Interbus Protocol Description](#) section is a low level description for those who wants to write their own code e.g. for a micro controller in their own system that should be able to communicate with NKT Photonics modules/systems.

NKTPDLL

The [NKTPDLL](#) section briefly explains about the DLL, and how to find additional information.

Examples

The [Examples](#) section briefly explains about the examples and how to use them.

LabVIEW Driver

The [LabVIEW Driver](#) section explains how to use the LabVIEW driver from the *LabVIEW\Labview Driver* folder.

Register Files

The [Register Files](#) section explains how to read and use the information from the register files from the *Register Files* folder. There are separate register files for the different types of modules and systems.

Silabs USB Driver

The USB driver for NKT Photonics products with USB interface (e.g. SuperK Extreme and Koheras AdjustiK), the USB driver can be found in the *Silabs* folder. The [Silabs USB Driver](#) section describes how to install the USB driver.

Generic User Interface

The Software Development Kit includes a Generic User Interface. This user interface uses the information from the register files in the 06_Register File folder, so it can be of help to e.g. understand what to write to which registers and to read out what has been written to the different registers. The [Generic User Interface](#) section describes how to install and use the Generic User Interface from the *Tools\Generic User Interface* folder.

2 Interbus Protocol Description

This section describes the NKT Photonics Interbus protocol and module hardware integration. It is not to be confused with INTERBUS developed by Phoenix Contact.

2.1 Physical

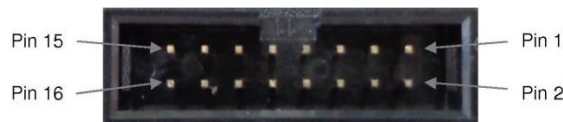
Physical

The NKT Photonics Interbus standard is based on the common RS-232 or 2-wire RS-485 interface. The serial port settings are: 115200 bits/s bit rate, 8 data bits, 1 stop bit, and no parity. In some systems, communication goes through a USB or Ethernet port, but the telegrams are built in the exact same way.

RS-485, modules

Single modules, that are designed to be combined with other modules on a shared bus, use RS-485 for communication. When communicating directly with these, e.g. through the “USB to RS485 adapter”, hardware handshake should be set to None.

The connector for this is a standard 16-pin ribbon cable connector (IDC, 2.54 mm pitch).

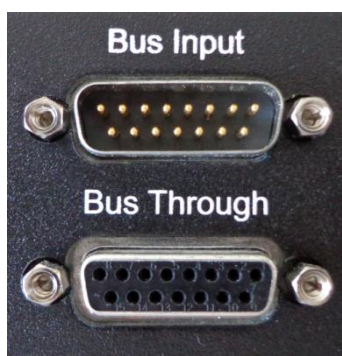


The IDC connector pinout

Pin #	I/O	Description
1	O	Module OK A high level (5 V) signal on this pin indicates that the module is running (emission on), and the optical output appears to be OK. In some modules, this pin is not connected.
2	O (I)	Emission LED On laser modules, this output supplies 5 V through a 300 – 330 Ω resistor when emission is on. On accessories without lasers, a positive voltage on this pin activates the local emission indicator.
3	I/O	RS-485 D– Inverted data signal
4	I/O	RS-485 D+ Non-inverted data signal
5	I	Interlock loop + Interlock loop signal input (5 V). This input is 12 V tolerant in most modules.
6	I	Module enable A high level (5 V) signal on this pin is necessary in order to turn emission on. This input is 12 V tolerant in most modules.
7	O	Interlock loop – Interlock loop signal output. The output on this pin goes high (5 V) when there is high level input on pin 5, and local interlock switches (in the module) are closed.
8	I	Interlock signal A high level (5 V) signal on this pin is necessary in order to turn emission on. This input is 12 V tolerant in most modules.
9..12		Ground
13..16	I	+12 V power supply

For 12 V tolerance information, please consult the individual module's user manual.

External accessory modules are not equipped with the IDC connector, but instead with a 15-pin D-sub connector. These modules have two connectors – male and female – to allow connecting several modules in a daisy-chain. The signals are the same, but the pinout is different.



The male D-sub 15 pinout:

Pin #	I/O	Description
1	O	Module OK A high level (5 V) signal on this pin indicates that the module is running (emission on), and the optical output appears to be OK. In some modules, this pin is not used.
2	I/O	RS-485 D- Inverted data signal
3	I	Interlock loop + Interlock loop signal input (5 V). This input is 12 V tolerant in most modules.
4	O	Interlock loop – Interlock loop signal output. The output on this pin goes high (5 V) when there is high level input on pin 3, and local interlock switches (in the module) are closed.
5..6		Ground
7..8	I	+12 V power supply
9	I (O)	Emission LED On accessories without lasers, a positive voltage on this pin activates the local emission indicator. On laser modules, this output supplies 5 V through a 300 – 330 Ω resistor when emission is on, and is high impedance when emission is off.
10	I/O	RS-485 D+ Non-inverted data signal
11	I	Module enable On laser modules, a high level (5 V) signal on this pin is necessary in order to turn emission on. This input is 12 V tolerant in most modules.
12	I	Interlock signal A high level (5 V) signal on this pin is necessary in order to turn emission on. This input is 12 V tolerant in most modules.
13..14		Ground
15	I	+12 V power supply

Please note, that in female connectors, the signal directions are opposite.

RS-485, master

Newer NKTP laser systems have a bus output that connects to the mentioned modules or accessories. This is always a female D-Sub 15 connector.

The bus output can supply a number of accessories with 12 V supply voltage. The absolute maximum current consumption is 4 A. The output is protected with a fuse, which cannot be replaced by customers. Also, the 12 V output is strictly an output. Do not attempt to supply light sources with 12 V through this port.



The master D-sub 15 (female) pinout:

Pin #	I/O	Description
1	(I)	Module OK return In most hosts, this pin is not used.
2	I/O	RS-485 D- Inverted data signal
3	O	Interlock loop + Interlock loop signal output. When the interlock circuit is closed, the nominal voltage is 5 V. However, when the circuit is open, there may be up to 12 V on this pin.
4	I	Interlock loop - Interlock loop signal return (5 V).
5..6		Ground
7..8	O	+12 V power supply
9	O	Emission LED This output supplies 5 V through a 300 – 330 Ω resistor when emission is on.
10	I/O	RS-485 D+ Non-inverted data signal
11	O	Laser system and emission OK Laser systems output 5 V when emission is on, and the system is OK.
12	O	Interlock signal This output supplies 5 V when the interlock circuit is closed, and the interlock relays are energized.
13..14		Ground
15	O	+12 V power supply

If an extra interlock circuit is necessary (apart from the standard interlock connector), pins 3 and 4 in this connector can be used. Please keep these signals isolated from other equipment, either by mechanical switches or by relays.

When the external bus is not used, the bus defeater must be connected, in order to close the interlock circuit.

The RS-485 signals in this connector are used only for communication between the laser system and accessories. Please do not attempt to communicate directly through this port. Instead, use the RS-232, USB or Ethernet port, whichever is available. If necessary, telegrams are relayed to the external bus by the laser system.

RS-232

Some systems are equipped with a standard RS-232 interface, in a standard 9-pin D-sub female connector.

When communicating directly with a single module (or a number of modules) on an RS-485 bus, there are no hardware handshake signals in use. However, when communicating with a complete system, through an RS-232 port, handshake signals are available. On the host, handshake mode must be set to either Hardware (CTS/RTS) or None. If None is used, please check that the host's RTS signal is set low (logic '0', positive voltage). Otherwise, the equipment may not be able to respond.



The RS-232 pinout

Pin #	I/O	Signal name	Host signal name
1		Not connected	CD
2	O	TxD	RxD
3	I	RxD	TxD
4		Not connected	DTR
5		Ground	Signal ground
6	O	DCR Constant +12 V through a 2 k Ω resistor	DCR
7	I	CTS	RTS
8	O	RTS	CTS
9		Not connected	RI
Shield		Ground	Ground

If the CTS (RTS) handshake signal on pin 7 is not connected, or is not used by the host, it must be bypassed. Otherwise, the light source is not able to respond. This can be achieved by connecting pin 7 directly to pin 6, and not to the host.

USB

Some products are equipped with a USB port for direct connection to a computer. This requires that a USB device driver is installed on the computer. This driver is present on the software CD-ROM that came with the product, for Windows only (in the 07_USB Driver folder). Alternatively, it can be downloaded from the USB device manufacturer's website, where it is available for Linux, Mac OSX and several versions of Windows.

<http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>

Although it is a USB port, it simulates a regular serial port (virtual COM port). Thus, the same serial port settings apply: 115200 bit/s bit rate, 8 data bits, 1 stop bit, and no parity.

When connecting to a complete system, we recommend that RTS/CTS handshake is enabled. However, when communicating with a single module through a USB to RS-485 adapter, hardware handshake must be set to None.

Ethernet

A few products are equipped with an Ethernet port. Presently, the used network protocol is UDP, the port number is 10001, and the fixed IP address is printed on a label on the equipment. The transmitted and received data, under the UDP, is exactly the same as with serial communication.

In future products, the network protocol and port number may change. Please contact NKT Photonics if you wish to use the Ethernet port.

2.2 Telegram

Framing

A telegram is a complete message framed by a start character and a stop character.

[SOT][MESSAGE][EOT]

Start Of Telegram (SOT): 13 0x0D

End Of Telegram (EOT): 10 0x0A

Message

A message consists of destination and source addresses, message type, register number, data bytes, and cyclic redundancy check.

Dest	Source	Type	Reg #	0..240 data bytes	CRC MSB	CRC LSB
------	--------	------	-------	-------------------	---------	---------

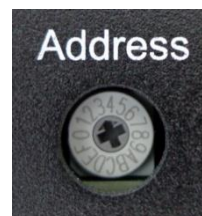
Destination address

The destination address specifies which module the message is intended for. Each module on a bus must have its own unique address, which is a number between 1 and 160. An older range was 1 .. 48, but for compatibility with future products, please change this.

Address 0 is reserved for special purposes. Addresses above 160 are considered host addresses (from PC or other equipment).

Most modules have a fixed, factory preset address, which cannot be changed by customers. For most of our products, these addresses can be found in chapter 6. Otherwise, please contact NKT Photonics to get information on specific module addresses.

Some external modules are designed to share a bus with several other external modules. These modules have an address selector, which is a rotary switch with 16 positions. The customer can use this to set the module address. The actual address is not really important, as long as no modules are set to the same address.



The switch position can be changed with a small screwdriver. A tiny arrow shows the setting (0..F), which corresponds to an address. The actual address is calculated by adding 16 (0x10) to the setting. In the picture above, the setting is "3", meaning that the address is 19 (0x13).

Even when communicating with a single laser system, the system may internally contain several modules, each with its own address. Therefore, the host may need to send telegrams to a number of different addresses on the same RS-232, USB or ethernet port.

Source address

The source address tells which module has sent the message, thus where the response should be directed. For host equipment (such as a computer), the source address must be greater than 160 (48 in older products). If you are making new interface software, please use a host address above 160 to avoid future address conflicts.

Message type

The message type (see table below) tells something about the purpose of the message.

Code	Type	Description
0	Nack	Response. Message not understood, not applicable, or not allowed (Not acknowledged).
1	CRC error	Response. CRC error in received message.
2	Busy	Response. Cannot respond at the moment. Module too busy.
3	Ack	Response. Received message understood (Acknowledged).
4	Read	Query. Read the contents of a register.
5	Write	Transmission. Write something to a register.
6	Write SET ¹	Transmission. Writing a logic one to a bit will set the corresponding bit in the register value. Logic zeros have no effect.
7	Write CLR1	Transmission. Writing a logic one to a bit will clear the corresponding bit in the register value. Logic zeros have no effect.
8	Datagram	Response. Register content returned; caused by a previous "Read".
9	Write TGL1	Transmission. Writing a logic one to a bit will invert (toggle) the corresponding bit in the register value. Logic zeros have no effect.

Messages from the host should usually be of type 4 or 5, or occasionally of type 6, 7 or 9. Normally, a module will respond with type 8 (datagram) if a type 4 message was sent, or 3 (acknowledge) if a type 5, 6, 7 or 9 message was sent.

Register number

The register tells the specific module what exactly the message is about. It consists of a single byte. Each module has its own list of registers, some of which are listed in chapter 6.

A few registers are standardized, meaning that they have the same purpose in all modules.

Reg #	Description
0x61	Module type number. Is used to determine what type of module the host is communicating with.
0x64	Firmware version code.

If the host tries to access a non-existing or restricted register, the module will answer with a Nack (not acknowledge, type 0) message.

Data bytes

The register byte can be followed by up to 240 data bytes. The meaning of these bytes depends on the type of module and the specific register. The contents vary a lot, but multi-byte values (more than one byte in size) are transmitted little-endian, i.e. LSB first.

CRC

The CRC value consists of two bytes calculated on the complete message. Contrary to the data bytes, the two CRC bytes are always transmitted big-endian, i.e. MSB first.

Prior to calculation, the CRC value must be set to 0. Below is an example on how to implement a CRC algorithm in C. Char is an 8 bit value, and int is a 16 bit value.

```
// Update the CRC for transmitted and received data using
// the CCITT 16bit algorithm (X^16 + X^12 + X^5 + 1).
```

¹ This message type is available only for a few specific registers, and is not usable in all modules.

```

unsigned char ser_data;
static unsigned int crc;

crc = (unsigned char)(crc >> 8) | (crc << 8);
crc ^= ser_data;
crc ^= (unsigned char)(crc & 0xff) >> 4;
crc ^= (crc << 8) << 4;
crc ^= ((crc & 0xff) << 4) << 1;

```

When transmitting, all bytes in the message must be passed one at a time through the CRC calculating function, and the resulting two CRC bytes are placed in the end of the message, MSB first. The serial data is entered in the variable `ser_data`, and the CRC result ends up in the variable `crc`.

When receiving, all bytes, including the CRC bytes, should be passed through the same CRC function. If there are no transmission errors, the CRC result is 0.

For software development purposes, this web page can be helpful for calculating and checking CRC values: <http://www.lammertbies.nl/comm/info/crc-calculation.html>. Remember to set the input type to Hex (not ASCII). The correct CRC result for Interbus is CRC-CCITT (XModem).

On-line CRC calculation and free library

- [Introduction on CRC calculations](#)
- [Free CRC calculation routines for download](#)
- [CRC calculation support forum](#) **New**

"0F420470" (hex)	
1 byte checksum	197
CRC-16	0x24A0
CRC-16 (Modbus)	0x00A0
CRC-16 (Sick)	0x08BD
CRC-CCITT (XModem)	0x1570
CRC-CCITT (0xFFFF)	0x91B0
CRC-CCITT (0x1D0F)	0x1B60
CRC-CCITT (Kermit)	0xD015
CRC-DNP	0xF59C
CRC-32	0x3E5022DC

Input type: ☐ ASCII ☒ Hex

Please note that the Lammertbies web site is not related to NKT Photonics in any way.

Telegram

When converting a message to a telegram, the message must first be scanned for special characters, which are 10 (0x0A), 13 (0x0D) and 94 (0x5E). These characters must be replaced with some predefined two-byte codes, which are explained in the next chapter.

After this is done, a 13 (0x0D) byte must be inserted in front of the message, a 10 (0x0A) byte must be added to the rear, and the telegram is ready to transmit.

Alternatively, the start-of-telegram byte, the end-of-telegram byte and the special character conversion can be part of the transmission routine.

Address cycling

In multi-tasking systems, it may sometimes be difficult to be certain which response fits to which transmitted telegram. Even though the NKT Photonics modules/systems

handle requests in the correct order, it is not always certain that the operating system or the software in the host computer does the same.

To overcome this, a simple solution can be to change the source address every time a new telegram is transmitted. When a module responds, the target address in the response telegram is identical to the source address in the request telegram, so the host computer can easily pair the response with the request. Addresses from 49 (0x31) to 255 (0xFF) can be used for this. In future products, the address space will start at 161 (0xA1).

Address scan

It is possible to make an address scan to see which module is found on which address. To do this, read register 0x61 from each possible module address, one at a time, with a 50 – 100 ms timeout. If a module is present on the address used, it will respond with its module type number to identify itself. If there is no response before timeout, there is no module present on that address.

Arrays

In some cases, the data bytes contains an array of separate values. In the register files and in this manual, these arrays are treated as zero-based (meaning that the first element is "number zero").

The byte order in arrays of 16-bit values is as shown:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Value 0		Value 1		Value 2		Value 3	
Bits 0-7	Bits 8-15	Bits 0-7	Bits 8-15	Bits 0-7	Bits 8-15	Bits 0-7	Bits 8-15

The byte order in arrays of 32-bit values is as shown:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Value 0				Value 1			
Bits 0-7	Bits 8-15	Bits 16-23	Bits 24-31	Bits 0-7	Bits 8-15	Bits 16-23	Bits 24-31

2.3 Special character conversion

Before a message is transmitted, special characters, if any, must be converted. If a data byte is equal to one of the three numbers 10, 13 or 94 (0x0A, 0x0D or 0x5E), they must be substituted with a two-byte word. The first substitution byte is always 94 (0x5E), and the second byte is the original byte where the value 64 (0x40) is added.

Name	Dec	Hex		Substitution
EOT	10	0x0A	End of telegram	5E 4A
SOT	13	0x0D	Start of telegram	5E 4D
SOE	94	0x5E	Start of substitution word	5E 9E

The reason for this conversion is that the SOT (13, 0x0D) and EOT (10, 0x0A) bytes can never ever be used for anything else than Start of Telegram and End of Telegram, in order to make telegram framing completely clear for all Interbus modules.

If a CRC byte is a special character, this must also be converted, as with any other byte. Even though the special character conversion seemingly changes the contents of a message, a new CRC value shall **not** be calculated after conversion.

Below is a C example on how to make the conversion work in transmission mode. The function "com1_putc" handles the transmission by the use of the UART.

```
void TX (unsigned char data)
{
    if(data == SOT || data == EOT || data == SOE)
    {
        com1_putc(SOE);
        data += 0x40;
    }
    com1_putc(data);
}
```

Example (all bytes in Hex):

When the following message (without framing) is transmitted:

[0A][42][05][32][0D][9C][F0],

the data bytes 0Ah and 0Dh must be converted, in order not to be interpreted as EOT and SOT bytes.

After conversion:

[5E][4A][42][05][32][5E][4D][9C][F0]

And the final telegram, including framing, will be:

[0D][5E][4A][42][05][32][5E][4D][9C][F0][0A]

When receiving a 94 (0x5E) byte, the receiver must discard this byte, and subtract 64 (0x40) from the next byte, in order to restore the original message.

2.4 Communication Examples

In the following examples, the host address is 162 (0xA2).

All values in brackets are hexadecimal bytes.

Example 1

In the first example, the operator wants to turn emission on in a SuperK Extreme light source. The module address is 15 (0x0F), the register is 0x30, and the data value is 3.

The basic message is put together like this:

0F	Destination address, which is the module address
A2	Host (source) address
05	Message type is "write"
30	Register number (Emission on/off)
03	Data value (meaning "all on")
BC	CRC most significant byte
E1	CRC least significant byte

So before the message is converted to a telegram, it looks like this:

[0F][A2][05][30][03][BC][E1]

Since there are no special characters in the message, it can be converted to a telegram directly:

[0D][0F][A2][05][30][03][BC][E1][0A]

If the light source has understood the message, and will comply, it will respond with an "acknowledge" answer:

[0D][A2][0F][03][30][48][2F][0A]

With framing removed:

[A2][0F][03][30][48][2F]

And each byte means:

A2	Destination address, which is the host address
0F	Source address, which is the module address
03	Message type is "acknowledge"
30	Register which is acknowledged*
48	CRC most significant byte
2F	CRC least significant byte

* In special cases, and in older modules, this byte may be 0.

Example 2

In the second example, the operator wants to set the power level in a K80-1 BasiK module. The destination address is 10 (0x0A), the register is 0x23, and the desired power level is 50 mW.

The power level resolution in the BasiK module is 10 μ W, so the value to transmit is 5000, meaning "5000 \times 10 μ W = 50 mW". The value 5000 is equal to 0x1388. So the least significant byte (LSB) is 0x88, and the most significant byte (MSB) is 13.

Thus, the basic message looks like this (special character is highlighted):

[0A][A2][05][23][88][13][3B][55]

The first byte is a special character, and must be converted to the corresponding two-byte substitution. Including framing, the complete telegram will be:

[0D][5E][4A][A2][05][23][88][13][3B][55][0A]

If the BasiK module has understood the message, and "Acknowledge" is activated (default turned off in the K80-1 BasiK module), it will respond with an "acknowledge" answer (special character substitute is highlighted):

[0D][A2][5E][4A][03][23][81][8D][0A]

With framing removed:

[A2][0A][03][23][81][8D]

Example 3

In the third example, the operator wants to read the fiber laser temperature from a K80-1 Basik module. The destination address is 10 (0x0A), and the register is 0x11. The readout is 37.214 °C (37214 m°C).

The basic message is put together like this:

0A	Destination address, which is the module address
A2	Host (source) address
04	Message type is "read"
11	Register number (measured fiberlaser temperature)
75	CRC most significant byte
83	CRC least significant byte

So before the message is converted to a telegram, it looks like this (special character is highlighted):

[0A][A2][04][11][75][83]

The first byte is a special character, and must be converted to the corresponding two-byte substitution. Including framing, the complete telegram will be:

[0D][5E][4A][A2][04][11][75][83][0A]

With the measured value above, the response from the Basik module looks like this (special character substitutes are highlighted):

[0D][A2][5E][4A][08][11][5E][9E][91][63][7E][0A]

With framing removed:

[A2][0A][08][11][5E][9E][91][63][7E]

And each byte means:

A2	Destination address, which is the host address
0A	Source address, which is the module address
08	Message type is "datagram"
11	Responded register
5E	Data value, least significant byte
91	Data value, most significant byte
63	CRC most significant byte
7E	CRC least significant byte

Put together, the data value is 0x915E, which equals 37214, which is the measured fiber laser temperature in m°C.

2.5 Code Example, Transmission

Below is a C-code example with all functions to transmit a complete telegram.

```
#define SOT 0x0D
#define EOT 0x0A
#define SOE 0x5E
#define ECC 0x40

// Calculate CRC value
unsigned int CRC_add1(char data, unsigned int crc)
{
    crc = (unsigned char)(crc >> 8) | (crc << 8);
    crc ^= data;
    crc ^= (unsigned char)(crc & 0xff) >> 4;
    crc ^= (crc << 8) << 4;
    crc ^= ((crc & 0xff) << 4) << 1;
    return crc;
}

// Send byte without calculating CRC value
void TXnocrc(char data)
{
    if(data == SOT || data == EOT || data == SOE)
    {
        com1_putc(SOE);    // Send SOE character to UART
        data += ECC;       // Convert special character
    }
    com1_putc(data);      // Send data to UART
}

// Send byte, and calculate CRC value
unsigned int tx1(char data, unsigned int crc)
{
    crc = CRC_add1(data, crc);
    if(data == SOT || data == EOT || data == SOE)
    {
        com1_putc(SOE);    // Send SOE character to UART
        data += ECC;       // Convert special character
    }
    com1_putc(data);      // Send data to UART
    return crc;
}

// TxTlg() transmits a complete telegram.
// dest is the destination address.
// size is number of data bytes to be transmitted.
// type is the message type byte.
// reg is the register number.
// *data is a pointer to the data.
// my_address is the address of the transmitter.
// DIR1 is a hardware pin to control direction of RS485 interface.

void TxTlg(char dest, char size, char type, char reg, char *data)
{
    unsigned int crc_value;
    char tt;

    DIR1 = TRANSMITTING;           // Change RS485 direction
    com1_putc(SOT);                 // Start telegram
    crc_value = tx1(dest, 0);        // Transmit destination address
    crc_value = tx1(my_address, crc_value); // Transmit source address
    crc_value = tx1(type, crc_value); // Transmit message type
    crc_value = tx1(reg, crc_value); // Transmit register number
```

```

for(tt=0;tt<size;tt++)
{
    crc_value = tx1(*data,crc_value);    // Transmit data
    data++;                             // Increment datapointer
}
TXnocr((crc_value>>8) & 0xFF);         // Transmit CRC MSB
TXnocr(crc_value & 0xFF);               // Transmit CRC LSB
com1_putc(EOT);                         // End telegram
DIR1 = RECEIVING;                      // Change RS485 direction
}

```

2.6 Code Example, Reception

Analysis of a received telegram is done in the reverse order of packing for transmission.

When a SOT is received a new telegram is initiated. Set CRC value to 0.

When a SOE is received, subtract 64 (0x40) from the next data byte and then calculate new CRC value. The SOE byte itself is not used in the CRC calculation.

When anything but SOT, EOT or SOE is received, calculate the new CRC.

When an EOT is received, the CRC value must be 0. Otherwise, a CRC-error has occurred.

For historical reasons, a few module types do not respond to "Write" commands. All future modules, however, will be able to answer a successful "Write" with "Acknowledge".

Below is a C-code example on how to implement the reception, reverse special character conversion and CRC check. Some additional checks may be implemented, to catch UART overrun errors etc.

```

_Bool TelegramReady, InFrame, SpecialChar; // boolean
char RxCounter, RxBuffer[BUFLNGTH];
unsigned int crc; // 16 bits

#define SOT 0x0D
#define EOT 0x0A
#define SOE 0x5E
#define ECC 0x40

void COM1_isr(void) // Called by UART interrupt
{
    char ch;
    while (UART1notEmpty)
    {
        if(!FERR1) // No framing error
        {
            ch = RxREG1; // Get byte from UART1
            switch(ch)
            {
                case SOT: // New telegram
                    RxCounter = 0;
                    InFrame = 1;
                    crc = 0;
                    TelegramReady = 0;
                    break;

                case EOT: // End of telegram
                    InFrame = 0;
                    if (crc == 0) TelegramReady = 1; // CRC ok and telegram ready
                    break;

                case SOE: // Start of special character
                    SpecialChar = 1;
                    break;

                default: // Message byte
                    if (InFrame == 1)
                    {
                        if (SpecialChar == 1) // If this byte is special...
                        {
                            SpecialChar = 0;
                            ch -= ECC; // ... subtract 0x40 from byte
                        }
                        crc = CRC_add1(ch, crc); // Calculate CRC value
                        if(RxCounter < BUFLNGTH) // If room in buffer...
                        {
                            RxBuffer[RxCounter] = ch; // ... copy byte to buffer
                            RxCounter++; // and increment index
                        }
                    }
                    break;
            }
        }
    }
}

```

3 NKTPDLL

This section briefly describes the NKTPDLL.

For further information please consult the *NKTPDLL Reference manual* located in the NKTPDLL folder.

Description

The NKTPDLL contains communication API with a full implementation of the NKT Photonics interbus protocol.

The DLL abstracts the protocol implementation and provides high level functions to read and write registers. The DLL is provided in both 32 and 64bit versions.

The API contains lightweight functions as the registerRead types, registerWrite types and the registerWriteRead types, supporting all register datatypes.

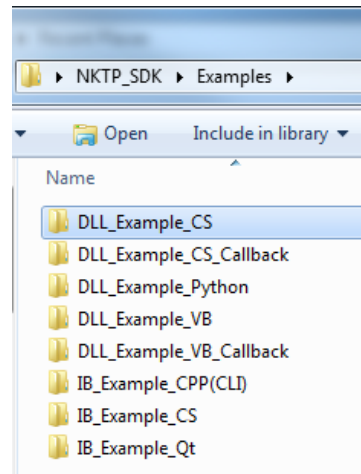
The API also contains a set of functions supporting advanced callback features, where the DLL kernel is constantly monitoring all the connected devices and their registers, generating callbacks to usercode when things change.

The *Examples* folder contains examples using the DLL in various environments.

The *LabVIEW/NKTP_API* folder contains examples using the DLL in LabVIEW.

4 Examples

The *Examples* folder.



The [DLL_Example_CS](#) folder contains an example written in C# using the DLL in a lightweight scenario.

The [DLL_Example_CS_Callback](#) folder contains an example written in C# using the DLL in a callback scenario.

The [DLL_Example_Python](#) folder contains various examples written in Python.

The [DLL_Example_VB](#) folder contains an example written in Visual Basic using the DLL in a lightweight scenario.

The [DLL_Example_VB_Callback](#) folder contains an example written in Visual Basic using the DLL in a callback scenario.

The [IB_Example_CPP\(CLI\)](#) folder contains an example written in C++(CLI) implementing the interbus protocol at a very basic level.

The [IB_Example_CS](#) folder contains an example written in C# implementing the interbus protocol at a very basic level.

The [IB_Example_Qt](#) folder contains an example written in C++ using the Qt framework, implementing the interbus protocol at a very basic level.

4.1 DLL_Example_CS & DLL_Example_VB

Purpose

This section describes the DLL_Example_CS and the DLL_Example_VB applications.

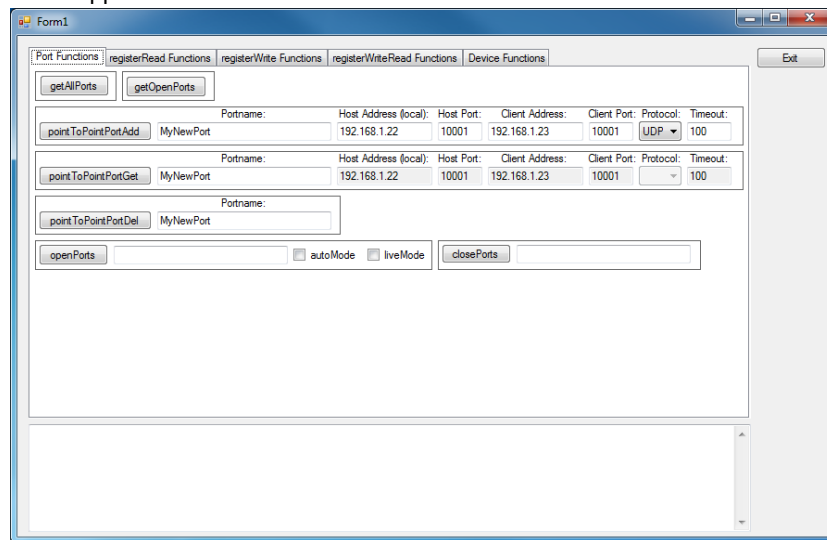
Requirements

The module(s)/system(s) to be controlled must feature a digital interface which is connected directly or via an adaptor to the computer. To compile and run the example code, you will need a Visual Studio installation.

The Visual Studio Community(express) edition could be downloaded from Microsoft's website.

Description

The DLL_Example_CS example is a small windows application written in C# source. The DLL_Example_VB example is a small windows application written in VB source. The example code illustrates how to interface and use the NKTPDLL to communicate and control the connected module(s)/system(s) using the lightweight functions available via the DLL. Almost all the lightweight functions are implemented in the applications.



Files

The NKTPDLL.cs/NKTPDLL.vb implements a class with the complete interface to the API.

Simply copy this file to your project folder and include the file in your project and implement your functionality.

The class will try to locate the correct version of the NKTPDLL.dll for your operating system, either x86(win32) or x64(win64) by using the windows environment variable "NKTP_SDK_PATH", the variable is initiated during the installation.

Alternatively you could copy the correct version into your release folder.

Note! – You might get the "LoaderLock" exception first time loading the NKTPDLL in the Visual Studio debugging mode. You could disable this warning by going to Debug-Exceptions-Managed Debugging Assistants and turn of the LoaderLock warning.

4.2 DLL_Example_CS_Callback & DLL_Example_VB_Callback

Purpose

This section describes the DLL_Example_CS_Callback and the DLL_Example_VB_Callback applications.

Requirements

The module(s)/system(s) to be controlled must feature a digital interface which is connected directly or via an adaptor to the computer. To compile and run the example code, you will need a Visual Studio installation.

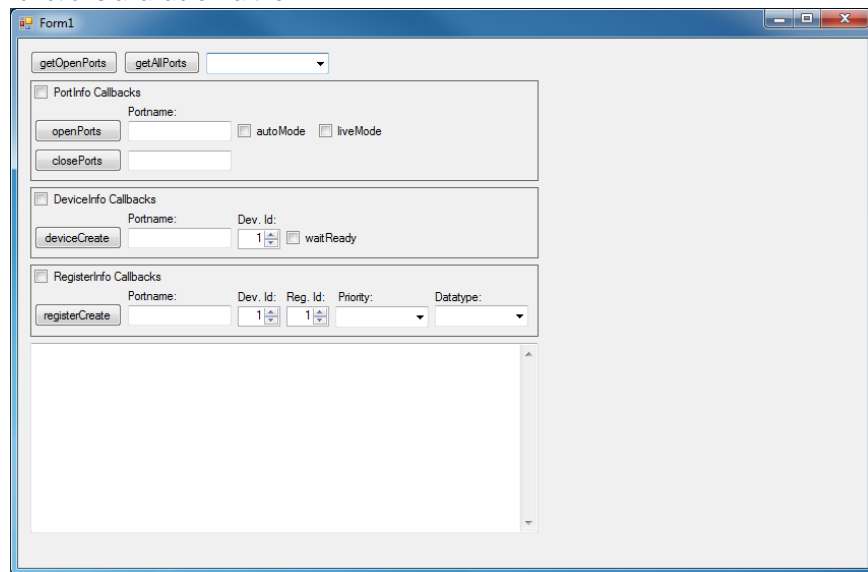
The Visual Studio Community(express) edition could be downloaded from Microsoft's website.

Description

The DLL_Example_CS_Callback example is a small windows application written in C# source.

The DLL_Example_VB_Callback example is a small windows application written in VB source.

The example code illustrates how to interface and use the NKTPDLL to communicate and control the connected module(s)/system(s) using the callback functions available via the DLL.



Files

The NKTPDLL.cs/NKTPDLL.vb implements a class with the complete interface to the API.

Simply copy this file to your project folder and include the file in your project and implement your functionality.

The class will try to locate the correct version of the NKTPDLL.dll for your operating system, either x86(win32) x64(win64) by using the windows environment variable "NKTP_SDK_PATH", the variable is initiated during the installation.

Alternatively you could copy the correct version into your release folder.

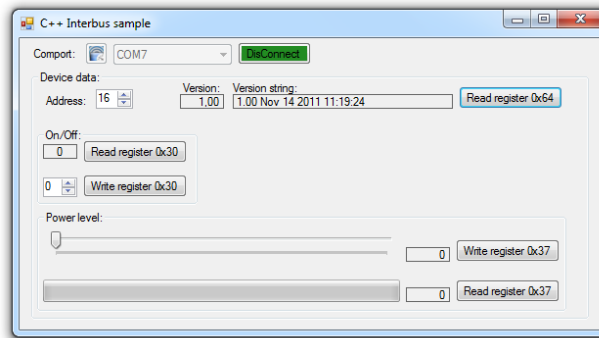
Note! – You might get the "LoaderLock" exception first time loading the NKTPDLL in the Visual Studio debugging mode. You could disable this warning by going to Debug-Exceptions-Managed Debugging Assistants and turn of the LoaderLock warning.

4.3 DLL_Example_Python

Purpose	This section describes the Python examples using the NKTPDLL.
Requirements	The module(s)/system(s) to be controlled must feature a digital interface which is connected directly or via an adaptor to the computer. To run the examples code, you will need to install Python which can be downloaded from python.org
Description	<p>All the python example scripts uses the script NKTP_DLL.py which contains the complete interface to the API using ctypes.</p> <p>The script NKTP_DLL.py will try to locate the correct version of the NKTPDLL.dll for your operating system, either x86(win32) x64(win64) by using the windows environment variable "NKTP_SDK_PATH", the variable is initiated during the installation.</p> <p>Alternatively you could copy the correct version into your script folder.</p> <p>Very basic and simple example using the NKTP_DLL.py script to turn on emission on an Extreme system, connected to COM27:</p> <pre>from NKTP_DLL import * result = registerWriteU8('COM27', 15, 0x30, 3, -1) print('Setting emission ON - Extreme:', RegisterResultTypes(result))</pre>

4.4 IB_Example_CPP(CLI) & IB_Example_CS

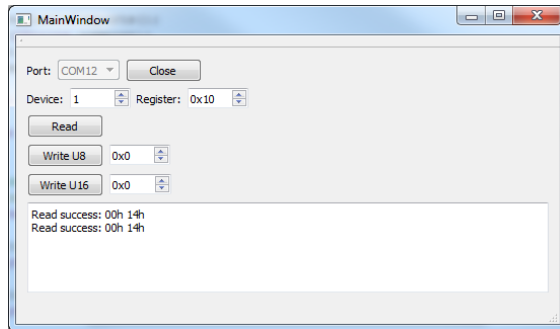
Purpose	This section describes the CPP(CLI) and the C# example application.
Requirements	<p>The module(s)/system(s) to be controlled must feature a digital interface which is connected directly or via an adaptor to the computer. To compile and run the example code, you will need a Visual Studio installation.</p> <p>The Visual Studio Community(express) edition could be downloaded from Microsoft's website.</p>
Description	<p>The IBSamplecpp example is a small windows application written in Visual Studio.</p> <p>The example code illustrates how to communicate and control the connected module(s)/system(s) via the NKT Photonics interbus protocol.</p>



Please note! The sample code does not implement retransmission on CRC error in the communication, 3-5 retries should be handled in a real world application. And ideally the communication should run in its own thread to allow the GUI to be responsive while the communication is in progress.

4.5 IB_Example_Qt (Using Qt framework)

Purpose	This section describes the C++ Qt example application.
Requirements	The module(s)/system(s) to be controlled must feature a digital interface which is connected directly or via an adaptor to the computer. To compile and run the example code, you will need a Qt installation. The Qt framework could be downloaded from the Qt website.
Description	The IBSampleQt example is a small windows application written in C++ source using Qt framework. The example code illustrates how to communicate and control the connected module(s)/system(s) via the NKT Photonics interbus protocol.



Please note! The sample code handles retransmission on CRC error in contrast to the C++/CLI sample which doesn't.

Files	<p>The sample is a complete self running sample, using the Qt framework. The folder <i>Interbus</i> contains an <i>Interbus.pri</i> file which is easily included in your own project. Simply copy the <i>Interbus</i> folder to your own project and include the <i>Interbus.pri</i> file in your own project file and include the <i>IBHandler.h</i> file where you need Interbus functionality.</p> <p>The <i>IBHandler</i> class contains a set of functions used to handle Interbus communication. Simply create an instance of the class, open a comport using the "openComport" function use the "writeRegister"/"readRegister" functions and when done use the "closeComport" function. See <i>mainwindow.cpp</i></p>
-------	---

5 LabVIEW Driver API

Description

This section describes how to use the NKT Photonics API functions from LabVIEW. The full API is documented in the “NKTPDLL Reference manual.pdf” that is linked in the “NKT Photonics->SDK” start menu. The LabVIEW API VIs are saved in LabVIEW version 2011, hence you can only use 2011 or later.

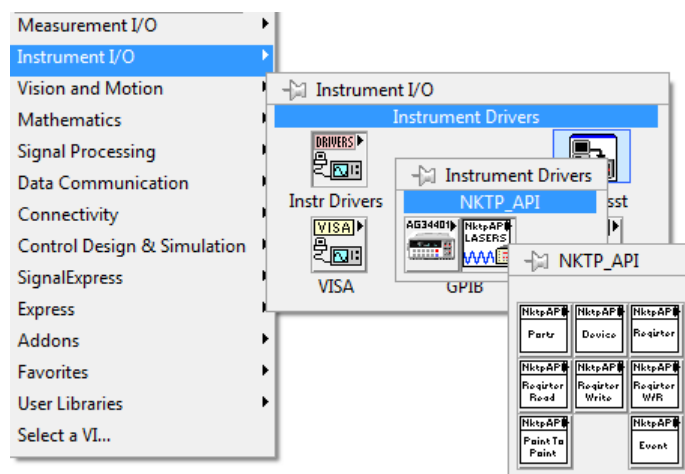
5.1 Adding the NKTP LabVIEW VIs to the LabVIEW palettes.

How to

We recommend that you copy the NKTP LabVIEW API VIs to the instr.lib folder in the LabVIEW versions that you use. The NKTP LabVIEW API VIs are located in the NKTP_API folder

“<Public Documents>\NKT Photonics\SDK\LabVIEW”.

After copying the NKTP_API folder and its content to the LabVIEW instr.lib folder and restarting LabVIEW, the NKTP palette will be available in the Instrument I/O block diagram palette, see below.



We recommend that you mass compile the NKTP_API VIs in the instr.lib location(s) after they have been copied there.

5.2 NKTP LabVIEW API structure

Description

The NKTP LabVIEW API is calling/wrapping the functions that are exported in the NKTP.dll. The NKTP.dll is globally registered so the LabVIEW VIs are able to find the DLL. The API has different classes of functions and the structure is shown in <NKTP_API\Public\NKTP_API_VITree.vi>, where NKTP_API is the LabVIEW instr.lib folder you have copied the API VIs to. Examples for using the API are included in LabVIEW instr.lib NKTP_API\Examples.

There is also included a VI template that can be used if you want to register LabVIEW User Events to asynchronous events from the NKTP.dll driver. Please consult the NKTP_API_UserEventTemplate.vit VI for further explanation.

Register data functions are included in polymorphic VIs.

The NKTP_API_VITree.vi block diagram contains all the API VIs and can be dragged and dropped to your own VIs for convenience and ease of use, see below.

Port Functions

NktpAPI
Open
Partnr

NktpAPI
Get All
Partnr

NktpAPI
Get Open
Partnr

NktpAPI
Close
Partnr

NktpAPI
Part
Error
Message

NktpAPI
Get
Legacy
Bur Scan

NktpAPI
Set
Legacy
Bur Scan

Device Functions

NktpAPI
Device
Create

NktpAPI
Device
Existnr

NktpAPI
Device
Get All
Typnr

NktpAPI
Device
Remove
All

NktpAPI
Device
Remove

NktpAPI
Device
Get
ErrorCond

NktpAPI
Device
Get
Type

NktpAPI
Device
Get
Made

NktpAPI
Device
Get
Live

NktpAPI
Device
Set
Live

NktpAPI
Device
Get
StatusBt

NktpAPI
Device
Get P/N
String

NktpAPI
Device
Get
Mod SN

NktpAPI
Device
Get
PCB SN

NktpAPI
Device
Get PCB
Version

NktpAPI
Device
Get
BL Ver

NktpAPI
Device
Get
BL VerSt

NktpAPI
Device
Get
FWVer

NktpAPI
Device
Get
FWVerSt

Register Functions

NktpAPI
Register
Create

NktpAPI
Register
Existnr

NktpAPI
Register
Get All

NktpAPI
Register
Remove

NktpAPI
Register
Remove
All

NktpAPI
ReqRead
(U8)

NktpAPI
ReqRead
(S8)

NktpAPI
ReqRead
(U16)

NktpAPI
ReqRead
(S16)

NktpAPI
ReqRead
(U32)

NktpAPI
ReqRead
(S32)

NktpAPI
ReqRead
(F32)

NktpAPI
ReqRead
(U64)

NktpAPI
ReqRead
(S64)

NktpAPI
ReqRead
(F64)

NktpAPI
ReqRead
(Arcii)

NktpAPI
ReqRead
(Void)

NktpAPI
ReqRead
(Void)

NktpAPI
ReqRead
(Void)

NktpAPI
ReqWrite
(U8)

NktpAPI
ReqWrite
(S8)

NktpAPI
ReqWrite
(U16)

NktpAPI
ReqWrite
(S16)

NktpAPI
ReqWrite
(U32)

NktpAPI
ReqWrite
(S32)

NktpAPI
ReqWrite
(F32)

NktpAPI
ReqWrite
(U64)

NktpAPI
ReqWrite
(S64)

NktpAPI
ReqWrite
(F64)

NktpAPI
ReqWrite
(Arcii)

NktpAPI
ReqWrite
(Void)

NktpAPI
ReqWrite
(Void)

NktpAPI
ReqWrite
(Void)

NktpAPI
ReqWrite
(U8)

NktpAPI
ReqWrite
(S8)

NktpAPI
ReqWrite
(U16)

NktpAPI
ReqWrite
(S16)

NktpAPI
ReqWrite
(U32)

NktpAPI
ReqWrite
(S32)

NktpAPI
ReqWrite
(F32)

NktpAPI
ReqWrite
(U64)

NktpAPI
ReqWrite
(S64)

NktpAPI
ReqWrite
(F64)

NktpAPI
ReqWrite
(Arcii)

NktpAPI
ReqWrite
(Void)

NktpAPI
ReqWrite
(Void)

NktpAPI
ReqWrite
(Void)

User Event Functions

NktpAPI
ReqUser
Eventnr

NktpAPI
ReqUser
Part
EventInd

NktpAPI
ReqUser
Device
EventInd

NktpAPI
ReqUser
ReqUser
EventInd

NktpAPI
ReqUser
Eventnr

NktpAPI
ReqUser
Data ta
LV Data

NktpAPI
ReqUser
Data ta
LV Data

NktpAPI
ReqUser
Data ta
LV Data

P2P Functions

NktpAPI
P2P
PartAdd

NktpAPI
P2P
Part Get

NktpAPI
P2P
Part Del

Void Void Void

6 Register Files

For information about what registers to read from and write to, please refer to the Register Files.

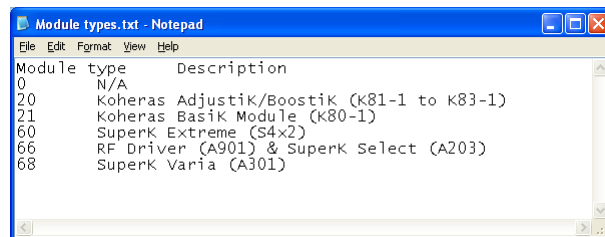
Each Register File provides information about read and read/write registers in one module or mainboard.

Notice

In systems like e.g. SuperK Extreme or Koheras BoostiK, it is recommended to communicate with the mainboard and not the different sub-modules, as it otherwise can happen that one or multiple sub-modules are operating in different states than the system mainboard is capable of.

Module types

Open the Module types.txt file with e.g. Microsoft Notepad to get an overview of the different types of modules.



So for communication with e.g. a SuperK Extreme system it is seen that it has the module type number 60h. So for information about registers to communicate with in a SuperK Extreme system, please refer to the register file "60.txt".

How to read register file

Open the desired register file in e.g. Microsoft Excel. The register files are tab delimited ASCII files, which Microsoft Excel can open and present.

Module type	Description
0	N/A
20	Koheras Adjustik/Boostik (K81-1 to K83-1)
21	Koheras Basic Module (K80-1)
60	SuperK Extreme (S4x2)
66	RF Driver (A901) & SuperK Select (A203)
68	SuperK Varia (A301)

Module type	Description	Value
60	SuperK Extreme (S4x2)	
#		
Readings		
11	NTC1 temperature	°C
#		
Controls		
30	Emission	0=Off;3=On
31	Setup bits	0=Current mode;1=Power mode
32	Interlock	(>0=reset interlock)
34	Pulse-Picker ratio	Times
35	Pulse-Picker delay	ns
36	Watchdog interval	Seconds
37	Power level	%
38	Current level	%
65	Module serial number	string
6C	User text	string
#		
Status bits		
0	Emission LED on	
1	Interlock off	
2	Interlock power failure	
3	Interlock loop off	
4	External disable	
5	Supply voltage low	
6	Module temp range	
7		
8		
9		
10		
11		
12		
13		
14	USB log error code present	
15	Error code present	
#		
Error code		
0	No error	
#		

First the file tells that in register 61h the module type can be read out from the module/system. In this case it will respond back with value 60h, which we from the Module types list know is a SuperK Extreme mainboard.

Readings and controls

The section with Readings includes all read only registers. The Controls section includes all registers that can be written to (unless they are write-protected). In most cases it is also possible to read the content of a Control register.

Register address

For Readings and Controls registers, their addresses are found in the first column.

Register description

In the second column there is a small description of the different registers.

Unit

The third column provides units for the different registers like: V, °C, Seconds, etc.

Type

The fourth column provides information about the type of content of the different registers,

Type	Variable
U8	Unsigned 8-bit Integer
U16	Unsigned 16-bit Integer
U32	Unsigned 32-bit Integer
I8	Signed 8-bit Integer
I16	Signed 16-bit Integer
I32	Signed 32-bit Integer
H8	Hexadecimal 8-bit Integer
H16	Hexadecimal 16-bit Integer
H32	Hexadecimal 32-bit Integer
string	Text string

Scaling

The last column tells what factor that should be multiplied with the contents of the actual register to get to the unit in the third column.

An example: A value 43264 is read from a register with the Unit °C, Type U16 and Scaling 0.001. The Type U16 tells it is an unsigned 16-bit integer, so it is a positive number. The Scaling factor 0.001 tells that the number has 3 digits after the decimal point, so 43264 becomes 43.264 °C.

6.1 General Registers

This section describes the common registers for NKT Photonics modules/systems.

- Module type (61h)** Register address 61h tells which module the host computer is communication with. This is used to establish or verify which type of module the host computer is connected to.
- Serial number (65h)** On register address 65h the serial number of the module/system can be read. The serial number is an 8 character string. In case of multiple modules/system on the same bus, the serial number can be used for identification of a given module/system.
- Status bits (66h)** On register address 66h the status of a module/system can be monitored. The status bits are combined in either one or more bytes. For explanation of the different status bits, please refer to the respective Register file.
- Error code (67h)** On register address 67h the module/system will provide an error code in case the actual module/system has shut down. The error code is read out as a byte value. Please refer to the respective Register file for more information about the various error codes.

Please notice that the Koheras AdjustiK/BoostiK Systems (K81-1 to K83-1) and Koheras BasiK Module (K80-1) do not generate error codes.

6.2 Koheras AdjustiK/BoostiK System (K81-1 to K83-1)

This section describes specific registers for Koheras AdjustiK/BoostiK Systems with product numbers starting with K81-1, K82-1 and K83-1.

Module type and module address	For communication with the Koheras AdjustiK/BoostiK System, communication should go through the mainboard. The mainboard module type number is 20h. The standard address is 0Fh (15 dec).
General	Output power and wavelength can be controlled for the system. The actual output power level and estimated wavelength cannot be monitored through the Interbus interface.
Setpoint (23h)	Dependant on whether the system is operating in current or power mode, the output power can be controlled by writing either a new pump current level or output power level to the setpoint register. The setpoint value is an unsigned 16-bit integer value. Pump current is set in mA, and output power is set in 1/100 mW (10 μ W).
Fiberlaser setpoint (25h)	<p>Dependant on whether the fiber laser is operating in temperature or wavelength mode, the wavelength can be controlled by writing either a temperature or wavelength to the fiberlaser setpoint register. The fiberlaser setpoint value is an unsigned 16-bit integer value. Temperature is set in milli-degrees C (1/1000 $^{\circ}$C). The wavelength setting is made in pm, and is relative to the wavelength offset value.</p> $\text{Total wavelength [nm]} = \text{Wavelength offset [nm]} + \text{Fiberlaser setpoint [pm]} / 1000$
Wavelength offset (28h)	The Wavelength offset is an unsigned 16-bit integer value of the wavelength of the laser in nanometers, 1-3 nm below the actual wavelength. The Wavelength is a constant value for a specific system as the wavelength adjustment is made with the Fiberlaser setpoint register.
Emission (30h)	The emission register is an unsigned 8-bit integer register. Writing 0 to this register will turn off laser emission from the Koheras AdjustiK/BoostiK System. Writing 1 to this register will turn on laser emission if the interlock circuit has been reset.

6.3 Koheras ADJUSTIK/ACOUSTIK System (K822 / K852)

6.3.1 General settings

This section describes specific registers for Koheras ADJUSTIK/ACOUSTIK Systems with product numbers starting with K822 or K852.

Module type and module address	For communication with the Koheras ADJUSTIK/ACOUSTIK System, communication should go through the mainboard. The mainboard module type number is 34h. The default address is 128.
Emission broadcast (30h)	Values written to this register will be broadcast to the emission on/off register in all laser modules. The value 1 turns emission on, and 0 turns emission off.
Broadcast setup (31h)	Values written to this register will be broadcast to the setup register in all laser modules. Please refer to the laser module register list in section 0.
Interlock (32h)	If the door interlock is in place, the key switch on the front plate is in On position and the External bus is terminated with e.g. a bus defeater, then the Interlock circuit can be reset via the Interbus interface by sending a value greater than 0 to the Interlock register.
Watchdog (34h)	The system can be set to make an automatic shut-off (laser emission only – not electrical power) in case of lost communication. The value in the watchdog register determines how many seconds with no communication the system will tolerate. If the value is 0, the feature is disabled. 8-bit unsigned integer. Maximum value is 255 seconds.
Broadcast wavelength offset (2Dh)	Values written to this register will be broadcast to the wavelength offset register in all laser modules.
Broadcast power, dBm (2Eh)	Values written to this register will be broadcast to the dBm power setpoint register in all laser modules.
Broadcast power, mW (2Fh)	Values written to this register will be broadcast to the linear (mW) power setpoint register in all laser modules.

6.3.2 Readouts

Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock relays off Bit 2: Interlock supply voltage low (possible short circuit) Bit 3: Interlock loop open Bit 4: Module address problem Bit 5: SD card problem Bit 6: Module communication problem Bit 7: No backplane detected Bit 8: Illegal MAC address Bit 9: Power supply low Bit 10: Temperature out of range ... Bit 15: System error code present</p>
Supply voltage (11h)	Internal supply voltage readout in mV. 16-bit unsigned integer.

6.3.3 Modulation, master module

Wavelength modulation frequency (22h)

This register is used for setting the internal wavelength modulation frequency. It is an array of two frequency values. Switching between these two frequencies is possible.

Each frequency value is a 32-bit integer, and the resolution is mHz. Frequency range is 8 mHz .. 100 kHz for sine and triangle waveforms, and 1 mHz .. 200 Hz for sawtooth and custom waveforms.

Wavelength modulation level (24h)

Wavelength modulation level. The resolution is tenths of percent (permille, ‰). 16-bit unsigned integer.

Wavelength modulation offset (25h)

Wavelength modulation offset. Adds a constant offset to the wavelength modulation signal, thus “pushing” the wavelength up or down. The resolution is tenths of percent (permille, ‰). 16-bit signed integer.

Amplitude modulation frequency (26h)

This register is used for setting the internal amplitude modulation frequency. It is an array of two frequency values. Switching between these two frequencies is possible.

Each frequency value is a 32-bit integer, and the resolution is mHz. Frequency range is 8 mHz .. 10 kHz for sine and triangle waveforms, and 1 mHz .. 200 Hz for sawtooth and custom waveforms.

Amplitude modulation max. power (28h)

Maximum peak level of modulation signal. The resolution is tenths of percent (permille, ‰). 16-bit unsigned integer.

Amplitude modulation depth (29h)

Amplitude modulation depth. The resolution is tenths of percent (permille, ‰). 16-bit unsigned integer.

Modulation setup (3Bh)

This register is used for modulation setup. 16-bit integer.

Bit 0: Amplitude modulation frequency selector.

Bit 1: -

Bit 2: Amplitude modulation source (0 = external; 1 = internal).

Bit 3: Enable amplitude modulation from main module (master) to internal laser modules.

Bits 4-6: Amplitude modulation waveform. See below.

Bit 7: -

Bit 8: Wavelength modulation frequency selector.

Bit 9: -

Bit 10: Wavelength modulation source (0 = external; 1 = internal).

Bit 11: Enable wavelength modulation from main module to internal laser modules.

Bits 12-14: Wavelength modulation waveform. See below.

Bit 15: -

Bit 6/14	Bit 5/13	Bit 4/12	Waveform
0	0	0	Sine
0	0	1	Triangle
0	1	0	Sawtooth (rising ramp)
0	1	1	Inverse sawtooth (falling ramp)
Other combinations			Reserved – do not use

6.3.4 Modulation, laser modules

Broadcast wavelength modulation on/off (3Eh)	<p>When the value 0 is written to this register, local wavelength modulation is turned off in all laser modules. When 1 is written, local wavelength modulation is turned on in all laser modules. 8-bit integer.</p> <p>This will automatically enable or disable the external modulation source in the laser modules.</p>
Broadcast amplitude modulation on/off (3Fh)	<p>When the value 0 is written to this register, local amplitude modulation is turned off in all laser modules. When 1 is written, local amplitude modulation is turned on in all laser modules. 8-bit integer.</p> <p>This will automatically enable or disable the external modulation source in the laser modules.</p>

6.3.5 Ethernet

IP address (B0h)	<p>Sets the system's IP address. An array of four 8-bit values.</p> <p>The byte order is the same as the normally written byte order. E.g. if the IP address is 192.168.1.17, the data byte order is [C0][A8][01][11].</p>
MAC address (B3h)	<p>The systems MAC address (read only). An array of six 8-bit values. The byte order is equivalent to the IP address.</p>
System port (B4h)	<p>The system's own TCP/UDP port number. 16-bit integer.</p>
Host port (B5h)	<p>If this value is zero, the module will respond to any sender. Otherwise, it will only respond to the port number set in this register. 16-bit integer.</p>
Host IP address (B7h)	<p>Host (computer) IP address. If this value consists of zeroes only, the module will respond to any sender. Otherwise, it will only respond to telegrams from a host with the IP address set in this register.</p>

6.3.6 Special options

Multichannel simulation (36h)	<p>Set this value to 0 for normal operation. 8-bit unsigned integer.</p> <p>This feature is useful for programmers who are making software for multichannel systems, but have only one laser module. When this value is not zero, the main module acts as if there are as many laser modules as the value dictates. The laser module on address 1 is "cloned", to make control software believe that there is more than one laser module.</p>
--------------------------------------	---

6.4 Koheras BasiK Module (K80-1)

This section describes specific registers for Koheras BasiK Modules with product numbers starting with K80-1.

Module type and module address	The module type number is 21h. The standard module address is 10 (0Ah), but the actual address may be different, as there can be several modules on one bus.
Emission (30h)	The emission register is an unsigned 8-bit integer register. Writing 0 to this register will turn off laser emission from the Koheras BasiK Module. Writing 1 to this register will turn on laser emission if the interlock circuit is closed.
Current/power mode (31h)	If the laser module supports both current and power mode, it is possible to shift between the two modes with the current/power mode register. The value 0 will put the laser module into current mode. The value 1 will put the laser module into power mode.
Setpoint (23h)	Dependant on whether the system is operating in current or power mode, the output power can be controlled by writing either a new pump current level or output power level to the setpoint register. The setpoint value is an unsigned 16-bit integer value. Pump current is set in mA and, output power is set in hundredths of milliwatt (resolution is 10 μ W).
Fiberlaser setpoint (25h)	Dependant on whether the fiber laser is operating in temperature or wavelength mode, the wavelength can be controlled by writing either a temperature or wavelength to the fiberlaser setpoint register. The fiberlaser setpoint value is an unsigned 16-bit integer value. Temperature is set in milli-degrees C. The wavelength setting is made in pm, and is relative to the wavelength offset value. $\text{Total wavelength [nm]} = \text{Wavelength offset [nm]} + \text{Fiberlaser setpoint [pm]} / 1000$
Piezo modulation (32h)	If the laser module features piezo modulation, this can be enabled and disabled with the piezo modulation register (32h). Writing the value 0 to the register will disable piezo modulation, whereas the value 1 will enable this. 8-bit integer.
RIN suppression (33h)	On E15 BasiK modules the RIN suppression circuit can be enabled or disabled by writing either 1 or 0 respectively to the RIN suppression register. 8-bit integer.
Temperature/wavelength mode (34h)	On Koheras BasiK Modules supporting both temperature and wavelength mode, it is possible to switch between these two modes by writing either 0 (temperature) or 1 (wavelength) to the temperature/wavelength mode register. 8-bit integer.
Temperature compensation mode (35h)	The BasiK modules have a temperature compensation that can compensate for changes in ambient temperature. This feature is switched on by writing a 1 to the temperature compensation register, and off by writing a 0 to that register. 8-bit integer.
Acknowledge mode (36h)	In BasiK modules, the acknowledge response can be switched on and off. This is done by writing 0 or 1 to the acknowledge mode register. The value 1 turns it on, and 0 turns it off. When acknowledge mode is off, the module will receive, but not respond to "write" telegrams. 8-bit integer.
Fiberlaser temperature (11h)	Fiberlaser temperature readout in m°C (1/1000 °C). 16-bit unsigned integer.
Pump current (15h)	Pump current readout in mA. 16-bit unsigned integer.
Output power (18h)	Output power readout in 1/100 mW. 16-bit unsigned integer.

Module temperature (19h)	Module Temperature readout in 1/10 °C. 16-bit signed integer.
Module input voltage (1Bh)	Module supply voltage readout in mV. 16-bit unsigned integer.
Wavelength readout (10h, index 12)	<p>Present wavelength readout in pm. This value is part of an array of 16-bit integer values.</p> <p>The value, which is an unsigned 16-bit integer, is not the actual wavelength, but is relative to the wavelength offset. To get the total wavelength, add the Wavelength offset this way:</p> $\text{Total wavelength [nm]} = \text{Wavelength offset [nm]} + \text{Wavelength readout [pm]} / 1000$
Wavelength offset (10h, index 13)	<p>The wavelength offset is an unsigned 16-bit integer value of the laser wavelength of in nanometers, 1-3 nm below the actual wavelength. The wavelength is a constant value for a specific system, as the wavelength adjustment is made with the fiberlaser setpoint register.</p> <p>The Wavelength offset value is used in conjunction with the Fiberlaser setpoint value or the Wavelength readout value.</p>

6.5 Koheras BASIK MIKRO Module (K0x2)

6.5.1 General settings

This section describes specific registers for Koheras BASIK MIKRO Modules with product numbers starting with K0x2.

Module type and module address	The module type number is 36h. The standard module address is 1 (01h), but the actual address may be different, as there can be several modules on one bus.
Emission (30h)	The emission register is an unsigned 8-bit integer register. Writing 0 to this register will turn off laser emission from the Koheras BASIK Module. Writing 1 to this register will turn on laser emission if the Interlock circuit is closed.
Setup bits (31h)	<p>Module setup bits. Please refer to product manual for a thorough explanation of these. 16-bit unsigned integer.</p> <p>Bit 0-7: Unused Bit 8: Pump operation constant current Bit 9-15: Unused</p> <p>It is possible to use Write SET, Write CLR and Write TGL on this register (see 2.2 Telegram). However, a firmware upgrade may be necessary.</p>
Output power setpoint (22h)	Output power setpoint in 1/100 mW. 16-bit unsigned integer.
Output power setpoint, dBm (A0h)	Output power setpoint in 1/100 dBm. 16-bit signed integer.
Wavelength offset setpoint (2Ah)	Used for adjusting the wavelength. Resulting wavelength is the sum of the standard wavelength and the wavelength offset. Resolution is 1/10 pm. Signed 16-bit integer.
User area (8Dh)	The user area is a 240-byte piece of non-volatile memory, where the user can store an ASCII text string or other type of data. This can be an identifier or any other form of information that should follow the light source.

6.5.2 Readouts

Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock off Bit 2: - Bit 3: - Bit 4: Module disabled Bit 5: Supply voltage low Bit 6: Module temperature out of range Bit 7: - Bit 8: - Bit 9: - Bit 10: - Bit 11: Waiting for temperature to drop Bit 12: - Bit 13: - Bit 14: - Bit 15: Error code present</p>
Output power (17h)	Output power readout in 1/100 mW. 16-bit unsigned integer.
Output power, dBm (90h)	Output power readout in 1/100 dBm. 16-bit signed integer.
Standard wavelength (32h)	Module wavelength when wavelength offset is 0. Resolution is 1/10 pm. 32-bit unsigned integer.
Wavelength offset (72h)	<p>Measured/calculated wavelength offset readout in 1/10 pm. 32-bit signed integer.</p> <p>To get the absolute wavelength, add the standard wavelength (register 32h).</p>
Module temperature (1Ch)	Module temperature readout in 1/10 °C. 16-bit signed integer.
Module supply voltage (1Eh)	Module supply voltage readout in mV. 16-bit unsigned integer.

6.6 Koheras BASiK Module (K1x2)

6.6.1 General settings

This section describes specific registers for Koheras Basik Modules with product numbers starting with K1x2.

Module type and module address	The module type number is 33h. The standard module address is 1 (01h), but the actual address may be different, as there can be several modules on one bus.
Emission (30h)	The emission register is an unsigned 8-bit integer register. Writing 0 to this register will turn off laser emission from the Koheras Basik Module. Writing 1 to this register will turn on laser emission if the Interlock circuit is closed.
Setup bits (31h)	<p>Module setup bits. Please refer to product manual for a thorough explanation of these. 16-bit unsigned integer.</p> <p>Bit 0: - Bit 1: Wide wavelength modulation range Bit 2: Enable external wavelength modulation Bit 3: Wavelength modulation DC coupled Bit 4: Enable internal wavelength modulation Bit 5: Enable modulation output Bit 6: - Bit 7: - Bit 8: Pump operation constant current Bit 9: External amplitude modulation source</p> <p>It is possible to use Write SET, Write CLR and Write TGL on this register (see 2.2 Telegram). However, a firmware upgrade may be necessary.</p>
Output power setpoint (22h)	Output power setpoint in 1/100 mW. 16-bit unsigned integer.
Output power setpoint, dBm (A0h)	Output power setpoint in 1/100 dBm. 16-bit signed integer.
Wavelength offset setpoint (2Ah)	Used for adjusting the wavelength. Resulting wavelength is the sum of the standard wavelength and the wavelength offset. Resolution is 1/10 pm. Signed 16-bit integer.
User area (8Dh)	The user area is a 240-byte piece of non-volatile memory, where the user can store an ASCII text string or other type of data. This can be an identifier or any other form of information that should follow the light source.

6.6.2 Readouts

Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock off Bit 2: - Bit 3: - Bit 4: Module disabled Bit 5: Supply voltage low Bit 6: Module temperature out of range Bit 7: - Bit 8: - Bit 9: - Bit 10: - Bit 11: Waiting for temperature to drop Bit 12: - Bit 13: - Bit 14: Wavelength stabilized (X15 modules only) Bit 15: Error code present</p>
Output power (17h)	Output power readout in 1/100 mW. 16-bit unsigned integer.
Output power, dBm (90h)	Output power readout in 1/100 dBm. 16-bit signed integer.
Standard wavelength (32h)	Module wavelength when wavelength offset is 0. Resolution is 1/10 pm. 32-bit unsigned integer.
Wavelength offset (72h)	<p>Measured/calculated wavelength offset readout in 1/10 pm. 32-bit signed integer.</p> <p>To get the absolute wavelength, add the standard wavelength (register 32h).</p>
Module temperature (1Ch)	Module temperature readout in 1/10 °C. 16-bit signed integer.
Module supply voltage (1Eh)	Module supply voltage readout in mV. 16-bit unsigned integer.

6.6.3 Modulation

Wavelength modulation frequency (B8h)

This register is used for setting the internal wavelength modulation frequency. It is an array of two frequency values. Switching between these two frequencies is possible.

Each frequency value is a 32-bit float, and the unit is Hz. Frequency range is 8 mHz .. 100 kHz for sine and triangle waveforms, and 8 mHz .. 200 Hz for sawtooth waveforms.

Wavelength modulation level (2Bh)

Wavelength modulation level. The resolution is tenths of percent (permille, ‰). 16-bit unsigned integer.

Wavelength modulation offset (2Fh)

Wavelength modulation offset. Adds a constant offset to the wavelength modulation signal, thus “pushing” the wavelength up or down. The resolution is tenths of percent (permille, ‰). 16-bit signed integer.

This feature is active only when internal modulation is enabled (register 31h, bits 3 and 4 set).

Amplitude modulation frequency (BAh)

This register is used for setting the internal amplitude modulation frequency. It is an array of two frequency values. Switching between these two frequencies is possible.

Each frequency value is a 32-bit float, and the unit is Hz. Frequency range is 8 mHz .. 10 kHz for sine and triangle waveforms, and 8 mHz .. 200 Hz for sawtooth waveforms.

Amplitude modulation depth (2Ch)

Amplitude modulation depth. The resolution is tenths of percent (permille, ‰). 16-bit unsigned integer.

Modulation setup (B7h)

This register is used for modulation setup. 16-bit integer.

Bit 0: Amplitude modulation frequency selector.

Bit 1: -

Bit 2: Amplitude modulation waveform. 0 = Sine. 1 = Triangle.

Bit 3: For future use only. Set to 0.

Bit 4: Wavelength modulation frequency selector.

Bit 5: -

Bits 6-7: Wavelength modulation waveform. See below.

Bit 7	Bit 6	Waveform
0	0	Sine
0	1	Triangle
1	0	Sawtooth (rising ramp)
1	1	Inverse sawtooth (falling ramp)

6.7 BoostiK OEM Amplifier (N83)

This section describes specific registers for a BoostiK OEM amplifier.

Module type and module address	The BoostiK OEM amplifier module type is 70h. The standard module address is 2 (02h).
Heat sink temperature (10h)	Heat sink temperature readout in 1/10 °C. 16-bit signed integer.
Supply voltage (12h)	Supply voltage readout in mV. 16-bit unsigned integer.
Output power readout (15h)	Optical output power readout in mW. 16-bit unsigned integer.
Amplifier temperature (17h)	Amplifier temperature readout in 1/10 °C. 16-bit signed integer.
Pump current readout (19h)	Amplifier pump current readout in mA. 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock off Bit 2: - Bit 3: - Bit 4: Module disabled Bit 5: Supply voltage low Bit 6: Module temperature out of range Bit 7: Pump temperature out of range Bit 8: Input power low Bit 9: Output power low Bit 10: Booster temperature alarm Bit 11: Booster pump temperature alarm Bit 12: Pooster pump bias alarm Bit 13: - Bit 14: - Bit 15: Error code present</p>
Pump current setpoint (21h)	Pump current setpoint when running the amplifier in constant current mode. The setpoint resolution is mA. 16-bit unsigned integer.
Output power setpoint (22h)	<p>Output power setpoint when running the amplifier in constant power mode.</p> <p>Note: The setpoint resolution is tenths of dBm (1/10 dBm). 16-bit signed integer.</p>
Off / On / Mode (30h)	<p>Register 30h is used for switching the amplifier on and off, and setting the operating mode. Set the value according to the list below.</p> <p>0: Amplifier off 1: Amplifier on, constant pump current 2: Amplifier on, constant output power</p>

6.8 SuperK EXTREME System (S4x2)

6.8.1 Main module

This section describes specific registers for SuperK EXTREME with product numbers starting with S4x2.

Module type and module address

The module type number is 60h. The standard module address is 15 (0Fh).

Inlet temperature (11h)

The inlet temperature readout 1/10 °C. 16-bit signed integer.

Emission (30h)

Emission on/off. The value 0 turns emission off, and the value 3 turns it on (if interlock circuit has been reset). 8-bit unsigned integer.

Reading the register returns the current emission state of the SuperK Extreme. Normally, it returns the same value as the one that was written. However, while switching emission on or off, the SuperK EXTREME will go through a small number of different states. So briefly, the returned value will differ from the one that was written.

Setup (31h)

With the Setup register, the operation mode of the SuperK EXTREME System can be controlled. The possible values are listed below; however, in some systems, not all modes are available. 16-bit unsigned integer.

- 0: Constant current mode
- 1: Constant power mode
- 2: Externally modulated current mode
- 3: Externally modulated power mode
- 4: External feedback mode (Power Lock)

Interlock (32h)

If the door interlock is in place, the key switch on the front plate is in On position and the External bus is terminated with e.g. a bus defeater then the Interlock circuit can be reset via the Interbus interface by sending a value greater than 0 to the Interlock register.

Additionally, the opposite function (switching interlock relays off) can be done by sending the value 0 to the interlock register.

Reading the interlock register returns the current interlock status, which consists of two unsigned bytes. The first byte (LSB) tells if the interlock circuit is open or closed. The second byte (MSB) tells where the interlock circuit is open, if relevant.

MSB	LSB	Description
-	0	Interlock off (interlock circuit open)
0	1	Waiting for interlock reset
0	2	Interlock is OK
1	0	Front panel interlock / key switch off
2	0	Door switch open
3	0	External module interlock
4	0	Application interlock
5	0	Internal module interlock
6	0	Interlock power failure
7	0	Interlock disabled by light source
255	-	Interlock circuit failure

Pulse picker ratio (34h)

For SuperK EXTREME Systems featuring the pulse picker option, the divide ratio for the pulse picker can be controlled with the pulse picker ratio register.

Note: When reading the pulse picker value, an 8-bit unsigned integer will be returned if the ratio is lower than 256, and a 16-bit unsigned integer otherwise. This is for historical reasons.

Watchdog interval (36h)	The system can be set to make an automatic shut-off (laser emission only – not electrical power) in case of lost communication. The value in the watchdog interval register determines how many seconds with no communication the system will tolerate. If the value is 0, the feature is disabled. 8-bit unsigned integer.
Power level (37h)	Power level setpoint in tenths of percent (permille, ‰). 16-bit unsigned integer.
Current level (38h)	Current level setpoint in tenths of percent (permille, ‰). 16-bit unsigned integer.
NIM delay (39h)	On systems with NIM trigger output, the delay of this trigger signal can be adjusted with the NIM delay register. The input for this register should be an unsigned 16-bit value from 0 to 1023. The range is 0 – 9.2 ns. The average step size is 9 ps.
Status bits (66h)	<p>This register returns the mainboard status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock relays off Bit 2: Interlock supply voltage low (possible short circuit) Bit 3: Interlock loop open Bit 4: Output Control signal low Bit 5: Supply voltage low Bit 6: Inlet temperature out of range Bit 7: Clock battery low voltage ... Bit 13: CRC error on startup (possible module address conflict) Bit 14: Log error code present Bit 15: System error code present</p>
User text (6Ch)	The SuperK EXTREME System has a User Text register. This is a 20 character ASCII string register, which can be used by the user for identification or other purposes. In the front panel menu, the operator can select to have the user text written on the top line in the display.

6.8.2 Front panel

Module type and module address	The module type number is 61h. The standard module address is 1 (01h).
Panel lock (3Dh)	If this register value is set to 1, the current or power level cannot be changed from the front panel. The panel lock feature is turned off by setting the register to 0, or by pressing the Cancel button on the front panel. 8-bit integer.
Display text (72h)	Readout of the text currently shown in the display. 80 ASCII bytes. Some characters (ASCII values 0 .. 7) are special characters.
Error flash (BDh)	If this register value is set to 1, the display backlight will flash on and off when an error occurs. 8-bit integer.

6.8.3 Booster

Module type and module address	The module type number is 65h. The standard module address is 5 (05h).
Booster emission runtime (80h)	The total time that the booster has been emitting light. The resolution is seconds. 32-bit unsigned integer.
Booster status bits (66h)	<p>This register returns the booster status bits. 16-bit integer. The list below is an exerpt.</p> <p>Bit 0: Emission on Bit 1: Interlock signal off Bit 2: Interlock loop input low Bit 3: Interlock loop output low Bit 4: Module disabled Bit 5: Supply voltage out of range Bit 6: Board temperature out of range Bit 7: Heat sink temperature out of range</p>

6.9 RF Driver (A901) for SuperK SELECT

This section describes specific registers for an Internal (as part of a SuperK EXTREME system) or External RF Driver (A901) together with a SuperK SELECT (A203) accessory. All communication is with RF Driver, not with SuperK SELECT.

Module type and module address	<p>The RF Driver module type is 66h. An Internal RF Driver, i.e. when mounted inside the SuperK Extreme chassis, the standard module address is 6. On an External RF Driver the address depends on the tiny rotary switch (marked 0..9 and A..F) on the rear. The module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.</p> <p>The SuperK SELECT module type is 67h. The address depends on the tiny rotary switch (marked 0..9 and A..F) located close to the connectors. The module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.</p>
RF power (30h)	RF Power out of the RF Driver to the SuperK SELECT is controlled with the RF Power register. Writing 0 to this register will turn off RF Power, whereas the value 1 will turn on RF Power if the RF Driver has its RF output connected to a RF input on a SuperK SELECT. 8-bit unsigned integer.
Setup bits (31h)	<p>Register 31h contains three setup bits which control the operation of the RF Driver.</p> <p>Bit 0: Use temperature compensation. When this feature is on, the RF Driver output is compensated for filter temperature.</p> <p>Bit 1: Use optimal power table. When this feature is on, RF amplitude settings are scaled according to wavelength settings.</p> <p>Bit 2: Blanking Level. Can be used to set the Blanking Level input high or low, without having an actual blanking level input signal.</p>
Minimum wavelength (34h)	The lowest usable wavelength with the connected crystal. Resolution is pm. 32-bit unsigned integer.
Maximum wavelength (35h)	The highest usable wavelength with the connected crystal. Resolution is pm. 32-bit unsigned integer.
Crystal temperature (38h)	The temperature of the connected crystal in 1/10 °C. 16-bit signed integer.
FSK mode (3Bh)	<p>A special non-free option. Permits the operator to switch quickly between different wavelengths. Default value is 0 (FSK off). Standard FSK mode is 3. 8-bit unsigned integer.</p> <p>Please contact NKT Photonics for further information.</p>
Daughter board enable/disable (3Ch)	<p>This enables and disables the ability for the RF Driver to modulate its RF channels. Setting the register to 0 will pass the controls to the Modulation inputs on the RF Driver. Setting the register to 1 will disable the Modulation inputs, and let the RF Driver's internal electronics control the RF channels.</p> <p>When using the external Control box (e.g. for FSK mode), this register must be set to 0. Default value is 1. 8-bit unsigned integer.</p>
Connected crystal (75h)	Index number of the connected crystal, in an 8-bit integer. The crystals are numbered consecutively, so the crystals in the SuperK SELECT with the lowest bus address are numbered 1 and 2, the crystals in the next SuperK SELECT are numbered 3 and 4, and so on.

If the returned value is 0, it means that no crystal is connected to the RF driver.

The value is read-only, and will change automatically when the RF cable is connected to or disconnected from SuperK SELECT RF-ports, or when the position of the RF switch in the SuperK SELECT is changed.

**Wavelengths
(90h .. 97h)**

The RF Driver features 8 channels, i.e. up to 8 wavelengths can be controlled at the same time. The first channel is controlled with register 0x90, the second is controlled with 0x91, and so on.

Each register holds a 4-element array of 32-bit unsigned integers, each setting a wavelength in pm. The four elements are for the four states of Frequency Shift Keying (FSK) operation (non-free option). If FSK mode is not used, only the element on index 0 is used.

When altering only the first element (index 0), it is OK to send only one 32-bit integer, and ignore the rest.

**Amplitudes
(B0h .. B7h)**

The amplitudes of the 8 channels in tenths of percent (permille, ‰). Register 0xB0 controls the amplitude for the first channel, register 0xB1 controls the amplitude for the second channel, and so on. 16-bit unsigned integers.

**Modulation gain
settings
(C0h .. C7h)**

For use with the non-free FSK option. Adjusts the gain of the analog modulation inputs. Each of the 8 channels has its own gain setting, thus using the registers C0h-C7h. Contact NKT Photonics for further information.

6.10 SuperK SELECT (A203)

This section describes specific registers for a SuperK SELECT.

Module type and module address	The SuperK SELECT module type is 67h. The address depends on the tiny rotary switch (marked 0..9 and A..F) located close to the connectors. The module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.
Monitor 1 readout (10h)	Readout from optional optical power monitor no. 1 in tenths of percent (permille, ‰). 16-bit unsigned integer.
Monitor 2 readout (11h)	Readout from optional optical power monitor no. 2 in tenths of percent (permille, ‰). 16-bit unsigned integer.
Monitor 1 gain (32h)	<p>Gain setting for optional optical power monitor no.1. There are eight gain levels, numbered 0..7, with 0 being the lowest gain level. Each level increase doubles the sensitivity. 8-bit unsigned integer.</p> <p>Note: Monitor gain settings should not be altered when the SuperK light source is running in external feedback mode (Power Lock).</p>
Monitor 2 gain (33h)	<p>Gain setting for optional optical power monitor no.2. There are eight gain levels, numbered 0..7, with 0 being the lowest gain level. Each level increase doubles the sensitivity. 8-bit unsigned integer.</p> <p>Note: Monitor gain settings should not be altered when the SuperK light source is running in external feedback mode (Power Lock).</p>
RF switch (34h)	<p>The SuperK SELECT is equipped with two RF input connectors, and one or two AOTF crystals. When switching from one crystal to the other, the operator may either unplug the RF cable and plug it into the other connector, or use the internal RF switch. When activated, the RF switch swaps the two RF connections. Set the register value to 0 for normal operation, or 1 to swap the crystal connections. 8-bit unsigned integer.</p> <p>Please note, that RF power must be off before the RF switch position is changed.</p>
Monitor switch (35h)	The SuperK SELECT can have up to two optical power monitors, but has only one monitor output connector. Use the Monitor switch register to select which power detector should be connected to the output connector. Set the register value to 0 to get the power from the first crystal, or 1 to get the power from the second crystal. 8-bit unsigned integer.
Crystal 1 minimum wavelength (0x90)	Returns the minimum usable wavelength in crystal 1. The value is an unsigned 32-bit integer, and the resolution is pm.
Crystal 1 maximum wavelength (0x91)	Returns the maximum usable wavelength in crystal 1. The value is an unsigned 32-bit integer, and the resolution is pm.
Crystal 2 minimum wavelength (0xA0)	Returns the minimum usable wavelength in crystal 2. The value is an unsigned 32-bit integer, and the resolution is pm.
Crystal 2 maximum wavelength (0xA1)	Returns the maximum usable wavelength in crystal 2. The value is an unsigned 32-bit integer, and the resolution is pm.

6.11 SuperK VARIA (A301)

This section describes specific registers for SuperK VARIA (A301) accessory.

Module type and module address	The SuperK VARIA module type is 68h. The address depends on the tiny rotary switch (marked 0..9 and A..F) located close to the connectors. The module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.
Monitor input (13h)	Current output power in tenths of percent (permille, ‰). Requires the optional monitor to be attached for this register content to be valid. 16-bit unsigned integer.
ND setpoint (32h)	The output level of the SuperK VARIA is controlled with an unsigned 16-bit integer value sent to the neutral density filter setpoint register. The value in this register is in tenths of percent (permille, ‰).
SWP setpoint (33h)	Short wave pass filter setpoint register in 1/10 nm. 16-bit unsigned integer.
LWP setpoint (34h)	Long wave pass filter setpoint register in 1/10 nm. 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: - Bit 1: Interlock off Bit 2: Interlock loop in Bit 3: Interlock loop out Bit 4: - Bit 5: Supply voltage low Bit 6: - Bit 7: - Bit 8: Shutter sensor 1 Bit 9: Shutter sensor 2 Bit 10: - Bit 11: - Bit 12: Filter 1 moving Bit 13: Filter 2 moving Bit 14: Filter 3 moving Bit 15: Error code present</p>

6.12 Extend UV (A351)

This section describes specific registers for Extend UV (A351) accessory.

Module type and module address	The module type number is 6Bh. The address depends on the tiny rotary switch (marked 0..9 and A..F) located close to the connectors. The module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.
Wavelength setpoint (31h)	Wavelength setpoint in 1/10 nm. 16-bit unsigned integer.
Maximum wavelength (32h)	Maximum selectable wavelength in 1/10 nm. 16-bit unsigned integer.
Minimum wavelength (33h)	Minimum selectable wavelength in 1/10 nm. 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: - Bit 1: Interlock off Bit 2: Interlock loop in Bit 3: Interlock loop out Bit 4: - Bit 5: Supply voltage low Bit 6: - Bit 7: - Bit 8: Shutter sensor 1 Bit 9: Shutter sensor 2 Bit 10: Shutter sensor 3 Bit 11: - Bit 12: Stepper 1 running Bit 13: Stepper 2 running Bit 14: - Bit 15: System error code present</p>

6.13 SuperK COMPACT (S024)

This section describes specific registers for SuperK COMPACT with product numbers starting with S024.

Module type and Module address	The module type number is 74h. The standard module address is 1 (01h).
Supply voltage (1Ah)	Internal supply voltage readout in mV. 16-bit unsigned integer.
Heat sink temperature (1Bh)	Heat sink temperature readout in tenths of °C (1/10 °C). 16-bit signed integer.
Trig level (24h)	Trig level setpoint in mV. 16-bit unsigned integer. Value range 0 .. 4095 mV.
Display backlight (26h)	Display backlight level in %. 16-bit unsigned integer. Value range 0 .. 100 %.
Emission (30h)	Emission on/off. The value 0 turns emission off, and 1 turns it on (if interlock circuit has been reset). 8-bit unsigned integer.
Trig mode (31h)	This register is used for setting the SuperK COMPACT operating mode, i.e. the pulse trig source. 8-bit unsigned integer.

- 0: Internal frequency generator
- 1: External trig
- 2: Software triggered burst
- 3: Hardware triggered burst
- 4: External gate on
- 5: External gate off

Interlock (32h)	If the door interlock is in place, the key switch on the front plate is in On position and the External bus is terminated with e.g. a bus defeater, then the Interlock circuit can be reset via the Interbus interface by sending a value greater than 0 to the Interlock register.
-----------------	---

Additionally, the opposite function (switching interlock relays off) can be done by sending the value 0 to the interlock register.

Reading the interlock register returns the current interlock status, which consists of two unsigned bytes. The first byte tells if the interlock circuit is open or closed. The second byte tells where the interlock circuit is open, if relevant.

MSB	LSB	Description
-	0	Interlock off (interlock circuit open)
0	1	Waiting for interlock reset
0	2	Interlock is OK
1	0	Front panel interlock / key switch off
2	0	Door switch open
3	0	External module interlock
4	0	Interlock power failure
255	-	Interlock circuit failure

Internal pulse frequency (33h)	This register sets the internally generated pulse frequency (repetition rate) in Hz. 32-bit unsigned integer. Value range: Minimum is always 1 Hz. Maximum is system dependant, and can be read from Maximum Internal Frequency register.
--------------------------------	---

Burst Pulses (34h)	This register sets the number of pulses per burst, when using trig mode 2 or 3. 16-bit unsigned integer. Value range 1 .. 65535.
--------------------	--

	When emission is on, in trig mode 2, writing to this register will initiate a burst. If the telegram contains a number of pulses, this value is stored before the burst is initiated. If no data bytes are present in the telegram, the previously set number of pulses is used.
Watchdog interval (35h)	The system can be set to make an automatic shut-off (laser emission only – not electrical power) in case of lost communication. The value in the watchdog interval register determines how many seconds with no communication the system will tolerate. If the value is 0, the feature is disabled. 8-bit unsigned integer.
Internal pulse frequency limit (36h)	Readout of maximum permissible internal frequency in Hz. 32-bit unsigned integer.
Power level (3Eh)	Power level setpoint in %. Values written to this register will be converted to repetition rates, which will overwrite the value in the Internal frequency register (33h). The calculated repetition rate is a percentage of the maximum internal frequency. 8-bit unsigned integer. Value range 0..100 %.
	Available in firmware version 1.04, and future versions.
Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock relays off Bit 2: Interlock supply voltage low (possible short circuit) Bit 3: Interlock loop open Bit 4: - Bit 5: Supply voltage low (internal 12 V supply – not mains voltage) Bit 6: Internal temperature out of range Bit 7: Pump temperature out of range Bit 8: Pulse overrun Bit 9: External trig signal level Bit 10: External trig edge detected ... Bit 15: System error code present</p> <p>When an external trig pulse (edge) is detected, bit 10 will be set for approximately 100 ms. Continuous trig edges, faster than 10 Hz, will make bit 10 be set continuously. About 100 ms after the last trig edge, bit 10 is cleared.</p>
Optical pulse frequency (71h)	Readout of pulse frequency (repetition rate) measured by the system. 32-bit unsigned integer. Measurement resolution is 2 Hz.
Actual internal trig frequency (75h)	Readout of the more ideally correct internal trig frequency. Since the internal frequency clock generator works by dividing a fixed frequency with an integer, not all frequencies can be generated. This register returns the actual (calculated) frequency with 1/100 Hz resolution. 32-bit unsigned integer.
Display text (78h)	Readout of the text currently shown in the display. 80 ASCII bytes. Some characters (ASCII values 0 .. 7) are special characters.
Power level (7Ah)	Readout of calculated power in %. This value is a readout of the measured repetition rate, relative to the maximum internal frequency, thus representing the relative average power. 8-bit unsigned integer.
	Available in firmware version 1.04, and future versions.

User area (8Dh)

The user area is a 240-byte piece of non-volatile memory where the user can store an ASCII text string or other type of information. This can be an identifier or any other form of information that should follow the light source.

6.14aeroPULSE (P000)

6.14.1 Control Unit

This section describes specific registers for the aeroPULSE control unit with product numbers starting with P000.

Module type and module address

The module type number is 71h. The standard module address is 15 (0Fh).

Inlet temperature (11h)

The inlet temperature readout 1/10 °C. 16-bit signed integer.

Emission (30h)

Emission on/off. The value 0 turns emission off, and the value 4 turns the aeroPULSE System with the first Amplifier Module on (if interlock circuit has been reset). If the aeroPULSE System features an additional amplifier based on aeroGAIN-ROD the value 5 turns everything on. 8-bit unsigned integer.

Reading the register returns the current emission state of the aeroPULSE. Normally, it returns the same value as the one that was written. However, while switching emission on or off, the aeroPULSE will go through a small number of different states. So briefly, the returned value will differ from the one that was written.

Setup (31h)

With the Setup register, the operation mode of the aeroPULSE System can be controlled. The possible values are listed below; 16-bit unsigned integer.

- 0: Constant current mode
- 1: Constant power mode
- 2: Externally modulated current mode
- 3: Externally modulated power mode
- 4: External feedback mode (Power Lock)

Interlock (32h)

If the door interlock is in place, the key switch on the front plate is in On position and the External bus is terminated with e.g. a bus defeater then the Interlock circuit can be reset via the Interbus interface by sending a value greater than 0 to the Interlock register.

Additionally, the opposite function (switching interlock relays off) can be done by sending the value 0 to the interlock register.

Reading the interlock register returns the current interlock status, which consists of two unsigned bytes. The first byte (LSB) tells if the interlock circuit is open or closed. The second byte (MSB) tells where the interlock circuit is open, if relevant.

MSB	LSB	Description
-	0	Interlock off (interlock circuit open)
0	1	Waiting for interlock reset
0	2	Interlock is OK
1	0	Front panel interlock / key switch off
2	0	Door switch open
3	0	External module interlock
4	0	Application interlock
5	0	Internal module interlock
6	0	Interlock power failure
7	0	Interlock disabled by light source
255	-	Interlock circuit failure

Watchdog interval (36h)

The system can be set to make an automatic shut-off (laser emission only – not electrical power) in case of lost communication. The value in the watchdog interval

	register determines how many seconds with no communication the system will tolerate. If the value is 0, the feature is disabled. 8-bit unsigned integer.
Power level (37h)	Power level setpoint in tenths of percent (permille, ‰). 16-bit unsigned integer.
Current level (38h)	Current level setpoint in tenths of percent (permille, ‰). 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the mainboard status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock relays off Bit 2: Interlock supply voltage low (possible short circuit) Bit 3: Interlock loop open Bit 4: Output Control signal low Bit 5: Supply voltage low Bit 6: Inlet temperature out of range Bit 7: Clock battery low voltage ... Bit 13: CRC error on startup (possible module address conflict) Bit 14: Log error code present Bit 15: System error code present</p>
User text (6Ch)	The aeroPULSE System has a User Text register. This is a 20 character ASCII string register, which can be used by the user for identification or other purposes.

6.14.2 Amplifier Module

The primary control of the Amplifier Module is handled via the Control Unit (see the previous section).

Module type and module address	The module type number is 6Ah. The aeroPULSE System can include up to two amplifier modules. The standard amplifier module address is 5 (05h) and the additional amplifier based on aeroGAIN-ROD has module address 6 (06h).
Output power (10h)	Output power readout in 1/1000 mW. 32-bit unsigned integer.
Reflection (11h)	Reflection readout in mW. 16-bit unsigned integer.
Input monitor (12h)	Input power readout in mW. 16-bit unsigned integer.
External feedback monitor (17h)	External feedback monitor in mV. 16-bit unsigned integer.
Current monitor (1Ah)	Pump current monitor in mA. 16-bit unsigned integer.
Power lock status (3Ah)	<p>Status readout of power lock. 8-bit integer.</p> <p>Bit 0: Power locking in progress Bit 1: Power locking locked Bit 2: Power locking failed</p>
Pump temperature (97h)	Pump temperature readout in 1/10 °C. 16-bit signed integer.

Maximum level reduction (B7h)	<p>For each time the pump current reach the driver current limit, the maximum level is reduced with 20%. The number of instances the driver reached its current limit can be read out from this register. 8-bit unsigned integer.</p> <p>To reset the maximum level reduction, write the value 0 to this register.</p>
DPL update interval (BFh)	Sample time for Digital Power Lock (PDL) in ms. 8-bit unsigned integer. Values can be from 10 ms to 65535 ms.
Booster status bits (66h)	<p>This register returns the amplifier module status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock signal off Bit 2: Interlock loop input low Bit 3: Interlock loop output low Bit 4: Module disabled Bit 5: External Bus voltage out of range Bit 7: Module temperature range Bit 8: Pump temperature range Bit 9: Reflection high Bit 10: Input power low Bit 11: Pump current to high Bit 15: Error code present</p>

6.15 SuperK EVO (S2x1)

This section describes specific registers for SuperK EVO with product numbers starting with S2x1.

Module type and Module address	The module type number is 7Dh. The standard module address is 15 (0Fh).																														
Base temperature (17h)	The Base temperature readout in tenths of °C (1/10 °C).16-bit signed integer.																														
24V supply (1Dh)	Readout of 24V supply in mV.16-bit unsigned integer.																														
External control input (94h)	Voltage monitoring of External control input in milli-Volts. 16-bit unsigned integer.																														
Output power setpoint (21h)	Operating setpoint in ‰ when laser is operating in Power mode. 16-bit unsigned integer. Value range 0..1000 ‰.																														
Current setpoint (27h)	Operating setpoint in ‰ when laser is operating in Current mode. 16-bit unsigned integer. Value range 0..1000 ‰.																														
Emission (30h)	Emission on/off. The value 0 turns emission off, and 2 turns it on (if interlock circuit has been reset). 8-bit unsigned integer.																														
Setup bits (31h)	<p>This register is used for setting the operating mode. 8-bit unsigned integer.</p> <p>0: Current mode 1: Power mode 2: Current mode with External enable (if available) 3: Power mode with External enable (if available) 4: Digital external power feedback (if available) 5: Analog external power feedback (if available)</p>																														
Interlock (32h)	<p>If the door interlock is in place, the key switch on the front plate is in On position and the External bus is terminated with e.g. a bus defeater, then the Interlock circuit can be reset via serial interface by sending a value greater than 0 to the Interlock register.</p> <p>Additionally, the opposite function (switching interlock relays off) can be done by sending the value 0 to the interlock register.</p> <p>Reading the interlock register returns the current interlock status, which consists of two unsigned bytes. The first byte tells if the interlock circuit is open or closed. The second byte tells where the interlock circuit is open, if relevant.</p> <table><tr><th>MSB</th><th>LSB</th><th>Description</th></tr><tr><td>-</td><td>00</td><td>Interlock off (interlock circuit open)</td></tr><tr><td>00</td><td>01</td><td>Waiting for interlock reset</td></tr><tr><td>00</td><td>02</td><td>Interlock is OK</td></tr><tr><td>10</td><td>00</td><td>Interlock power failure</td></tr><tr><td>20</td><td>00</td><td>Internal interlock</td></tr><tr><td>30</td><td>00</td><td>External Bus interlock</td></tr><tr><td>40</td><td>00</td><td>Door interlock</td></tr><tr><td>50</td><td>00</td><td>Key switch</td></tr><tr><td>FF</td><td>-</td><td>Interlock circuit failure</td></tr></table>	MSB	LSB	Description	-	00	Interlock off (interlock circuit open)	00	01	Waiting for interlock reset	00	02	Interlock is OK	10	00	Interlock power failure	20	00	Internal interlock	30	00	External Bus interlock	40	00	Door interlock	50	00	Key switch	FF	-	Interlock circuit failure
MSB	LSB	Description																													
-	00	Interlock off (interlock circuit open)																													
00	01	Waiting for interlock reset																													
00	02	Interlock is OK																													
10	00	Interlock power failure																													
20	00	Internal interlock																													
30	00	External Bus interlock																													
40	00	Door interlock																													
50	00	Key switch																													
FF	-	Interlock circuit failure																													
Watchdog timer (36h)	The system can be set to make an automatic shut-off (laser emission only – not electrical power) in case of lost communication. The value in the watchdog interval																														

register determines how many seconds with no communication the system will tolerate. If the value is 0, the feature is disabled. 8-bit unsigned integer.

NIM delay (3Bh)	The NIM trigger output can be adjusted in time with the NIM delay register. The input for this register should be an unsigned 16-bit value from 0 to 1023. The range is 0 – 9.2 ns. The average step size is 9 ps.
Status bits (66h)	<p>This register returns the status bits. 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock relays off Bit 2: Interlock supply voltage low (possible short circuit) Bit 3: Remote interlock Bit 4: - Bit 5: Supply voltage low Bit 6: System temperature out of range ... Bit 14: Log error Bit 15: System error code present</p>
User area (8Dh)	The user area is a 240-byte piece of non-volatile memory where the user can store an ASCII text string or other type of information. This can be an identifier or any other form of information that should follow the light source.
IP address (B0h)	<p>Sets the system's IP address (static). An array of four 8-bit values.</p> <p>The byte order is the same as the normally written byte order. E.g. if the IP address is 192.168.1.17, the data byte order is [C0][A8][01][11].</p>
Gateway (B1h)	IP address for the gateway. An array of four 8-bit values. The byte order is equivalent to the IP address.
Subnet mask (B2h)	Subnet mask. An array of four 8-bit values. The byte order is equivalent to the IP address.
MAC address (B3h)	The systems MAC address (read only). An array of six 8-bit values. The byte order is equivalent to the IP address.

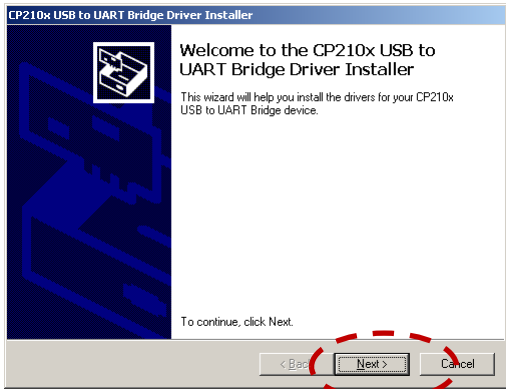
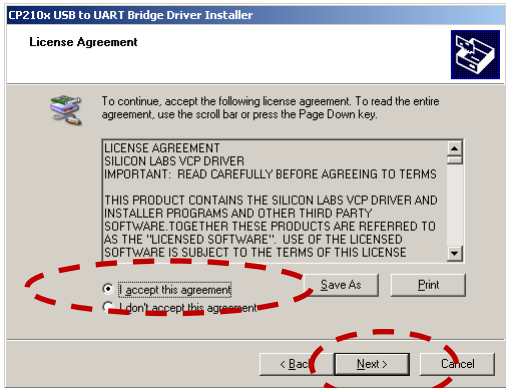
7 Silabs USB Driver

Install the Silicon Laboratories CP210x USB to UART Bridge software before connecting the system to the USB port on the computer.

USB Driver

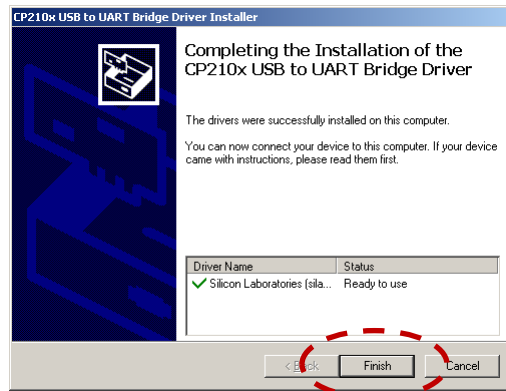
The following procedure describes how to install the driver software.

note! The driver is installed by the SDK installer by default, but if you opted not to install it during the SDK install, you have to install it according to this guideline.

Step	Description
1	The USB driver is found in the <i>Silabs</i> folder in the Software Development Kit.
2	<p>If the driver is installed on a windows Vista or XP system, run the installer: Silabs\670\CP210xVCPInstaller_x86.exe for 32bit or Silabs\670\CP210xVCPInstaller_x64.exe for 64bit systems.</p> <p>If the driver is installed on a windows 7, 8 or 10, run the installer: Silabs\674\CP210xVCPInstaller_x86.exe for 32bit or Silabs\674\CP210xVCPInstaller_x64.exe for 64bit systems.</p>
3	<p>Click Next on the window that appears.</p> 
4	<p>Read the License Agreement, choose "I accept this agreement", and click Next.</p> 

5

When the installation is complete, click Finish to exit the installer.



8 Generic User Interface

Description

The Generic User Interface software is a development tool, i.e. a help for programmers to verify and understand how to communicate with NKT Photonics products. The Generic User Interface is capable of controlling all NKT Photonics products communicating via the NKT Photonics Interbus, e.g. Koheras BasiK module and SuperK EXTREME products on a PC running Microsoft Windows.

LabVIEW Runtime Engine

The Generic User Interface is developed in LabVIEW 2011, which means that the LabVIEW 2011 runtime engine will be installed with the Generic User Interface.

8.1 Installing the software

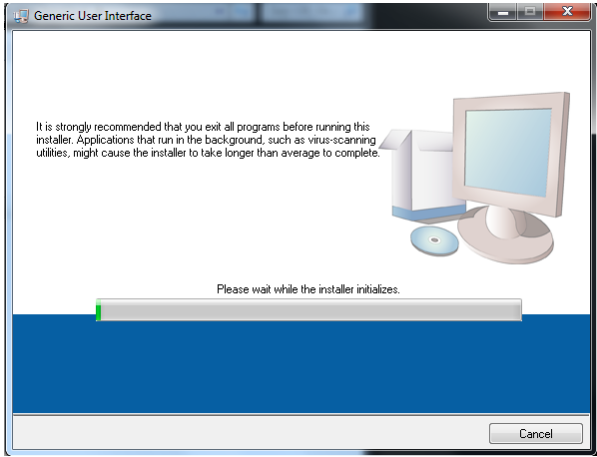
USB and Ethernet

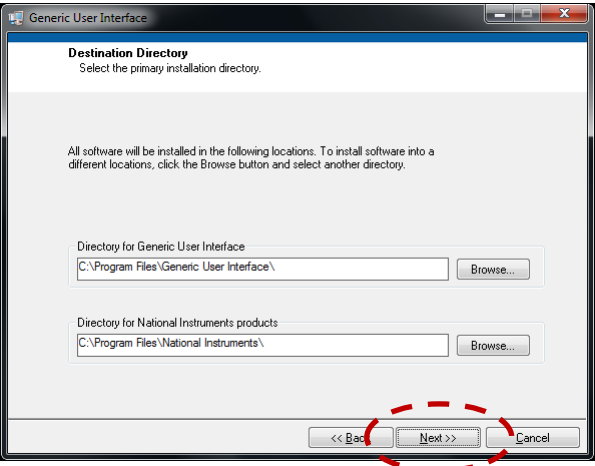
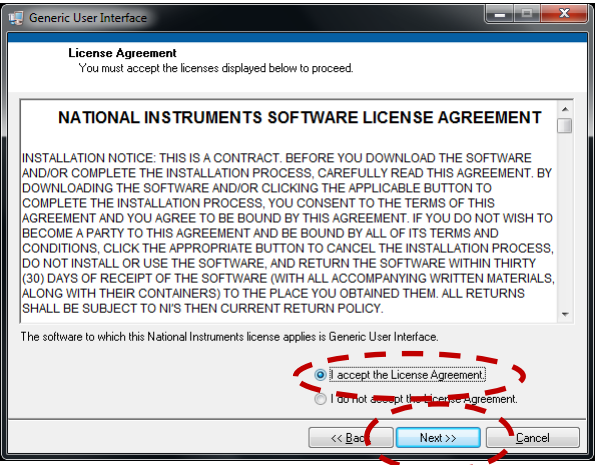
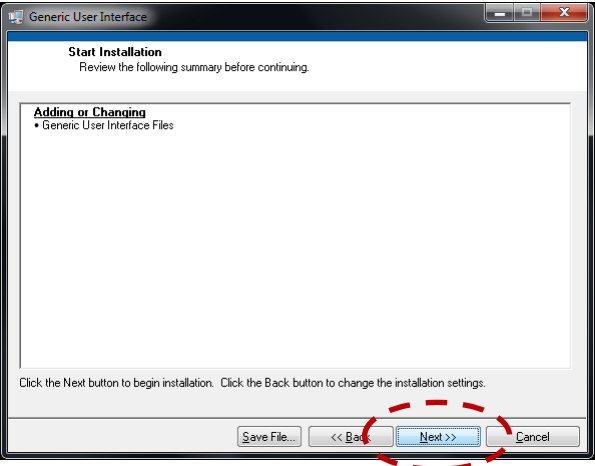
For information about how to install USB driver please refer to [Silabs USB Driver](#)
For information about how an Ethernet interface should be configured for communication, please refer to the hardware instruction manual.

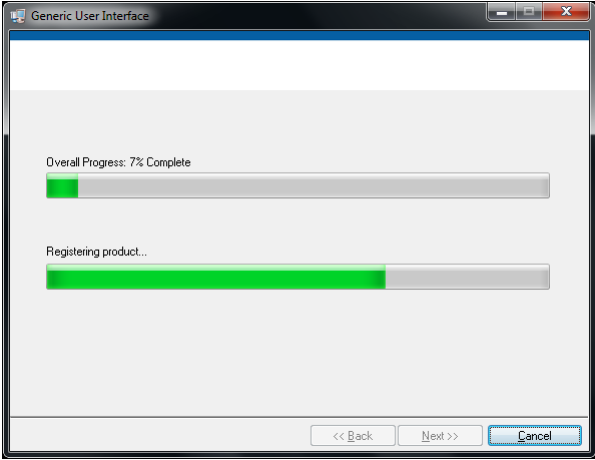
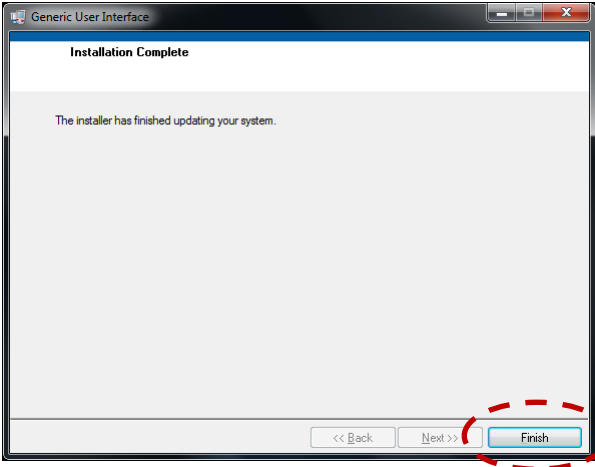
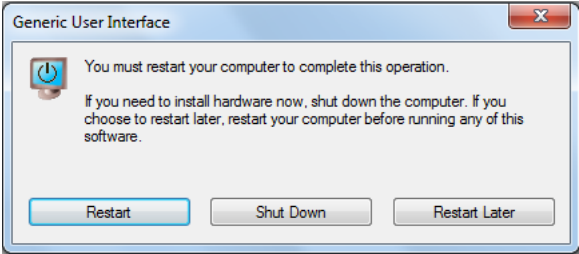

Generic User Interface Software

Use the following procedure to install the Generic User Interface software.

note! The Generic User Interface is installed by the SDK installer by default, but if you opted not to install it during the SDK install, you have to install it according to this guideline.

Step	Description
1	The installation package is located in the <i>Tools\Generic User Interface</i> folder in the Software Development Kit.
2	Run setup.exe.
3	<p>An installer window will show up on the screen.</p> 

4	<p>The installer asks for destination directories. Choose the suggested directories by clicking Next.</p> 
5	<p>Read the License Agreement, and choose I accept the License Agreement(s) and Next to proceed.</p> 
6	<p>Click Next to start copying files to the computer.</p> 

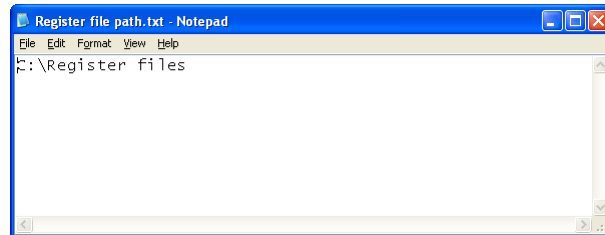
7	<p>A new window shows the progress of the installation.</p> 
8	<p>When the installation has completed, click Next to end the installer.</p> 
9	<p>After installation of the software, the computer may have to be restarted. In this case click on the Restart button to restart the computer.</p> 
10	<p>In the Start, Programs folder you will find a shortcuts to Generic User Interface. Click on this shortcut to run Generic User Interface.</p> 

8.2 Register File Path

The Generic User Interface uses the Register Files described in [Register Files](#). When the Generic User Interface has found a module on the chosen communication port and module address it will determine its module type number. The responded module type number will then be used to locate the corresponding register file, and from this it will show control and reading registers.

In the file folder where the Generic User Interface.exe file is installed, there is a file called "Register file path.txt". Open this file and make sure the path is correct for the folder holding the Register files described in [Register Files](#).

The default register file path is C:\Register files, but change this in the text file if the files are located another place.



8.3 Using the Generic User Interface Software

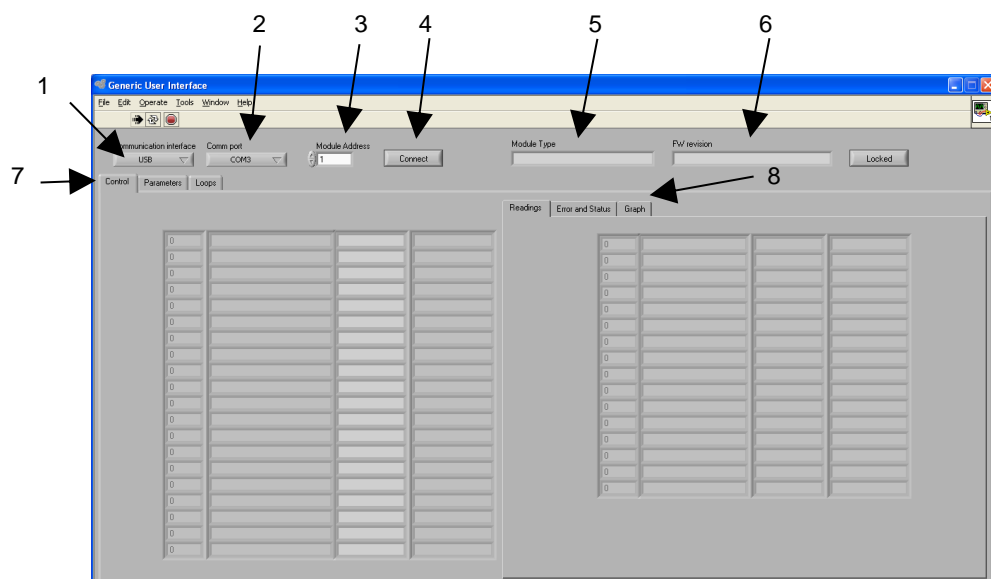
Introduction

This section includes 8 parts, representing the overall control of the user interface:

- **Front Panel : General introduction to the user interface.**
- **Starting Up: How to start up and use the user interface.**
- **Controls: Read/Write registers.**
- **Readings : Read registers.**
- **Error and Status: Error Code and Status Bits.**
- **Graph: Graphical view of register values.**
- **Functions: Upload firmware and download log.**
- **Loops: Loop sequence.**

8.3.1 Front Panel

Below the primary buttons and functions are described for the front panel of the Generic User Interface.



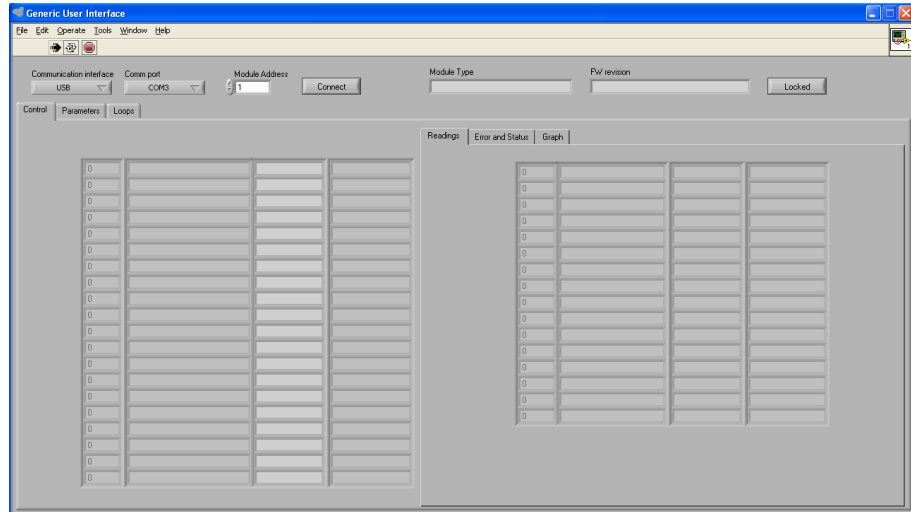
1 – Communication Interface	Choose between USB or Ethernet.
2 – Comm Port or IP Address	<p>For USB, choose the COM-port to communicate on. COM-port marked with *, indicates a known type of communication port, so this is likely the COM-port to choose.</p> <p>For Ethernet, type in the IP address to communicate with.</p>
3 – Module Address	Type in (or use the up/down buttons to select) the module address to communicate with.
4 - Connect	<p>Click on the Connect button to try establish communication between the Generic User Interface and a module/system on the chosen Module Address on the chosen Communication Port.</p> <p>When the Connect button is active, it says Connected. Click again to end and close the communication port.</p>
5 – Module Type	In this text field the Generic User Interface will inform about what module type it is communicating with. This text string comes from the Register File for the actual type of module.
6 – Firmware Revision	In this text field the user interface will inform about the current firmware revision in the module it is communicating with.
7 - Primary Panes	<p>In the upper left corner it is possible to shift between the primary panes of the Generic User Interface.</p> <p>Controls:</p> <ul style="list-style-type: none"> This pane includes all input registers including readouts of output registers. See Controls and Readings for details. <p>Functions:</p> <ul style="list-style-type: none"> On this pane it may be possible to upload new firmware to the module and download log data. See Functions for details. <p>Loops:</p> <ul style="list-style-type: none"> On this pane it is possible to automatically step through a sequence of register values. See Loops for details.
8 - Secondary Panes	With the Primary Pane set to Controls it is possible to chose between different other views of register values. See Error and Status and Graph for details.

8.3.2 Starting Up

Starting Up

Start Generic User Interface via Start - Programs – Generic User Interface or with the Generic User Interface short-cut on the desktop.

Choose between USB or Ethernet as Communication interface.

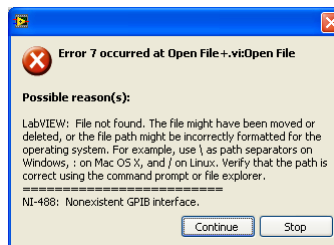


If USB is used, then select the COM port number for the connection (an asterisk next to the COM port in the drop down menu indicates that it is a known Communication interface, so this is likely the COM port to chose). If Ethernet is used, then type in the IP address for the system to communicate with.

Chose the Module Address to communicate with. The Module Address can be changed on-the-fly even if the Generic User Interface is already connected.

Register File Path

Make sure the correct path to the [Register Files](#) are defined in the [Register File Path](#) file. If the Generic User Interface cannot find the register files it will not be able to establish communication and the dialog below will appear.



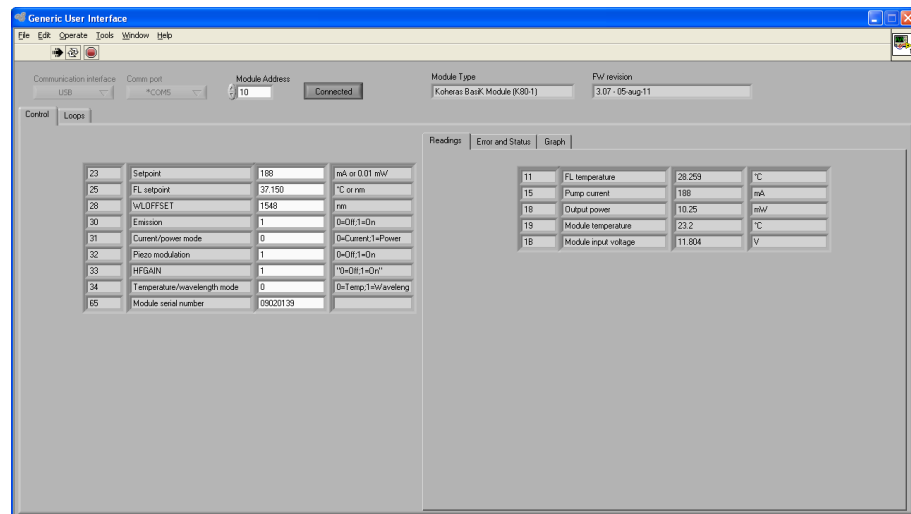
Click Stop and define the correct path in the Register file path.txt file as described in the [Register File Path](#) section.

Click on the Connect button to try to establish communication between computer and a NKT Photonics module/system.

If communication cannot be established, then verify the correct Communication Port and Module Address is used and that the Path field points at the directory with the Register Files.

8.3.3 Controls

In the left side of the Controls window it is possible to read out and write new values to input registers.



The first column of the array informs about the hexadecimal number for the register address. The second column provides a small description of the register and the fourth column provides the unit for the different registers. The content of these three columns is taken directly from the actual Register File.

Register Values

The third column provides the read out from the different registers. The Generic User Interface handles the scaling of the different registers.

Notice

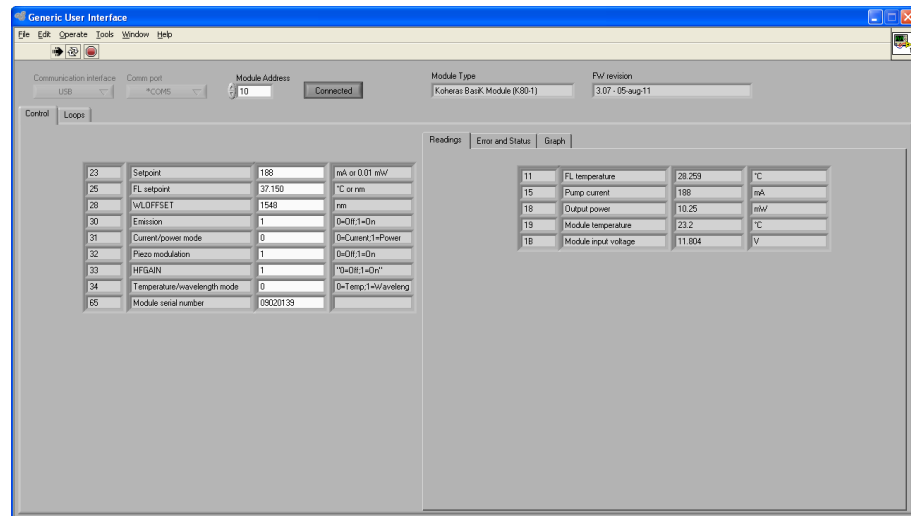
The third column is a string array, even if values are numbers. So the Enter button on the key board is not always the right thing to use when a new value has been typed into the array. Instead click with the mouse on somewhere on the grey back ground for the value to be written to the module.

Warning

If the Module Address is changed on-the-fly, be sure that the content of the array has been updated with the actual content from the new module before a new value is send to the module. Otherwise it may happen that other settings from the previous module are send to the new module. It usually takes a few seconds from the Module Address is changed to the array has been updated when the Minimum Sample Time is set to 1 second (see [Graph](#) for details).

8.3.4 Readings

On the Controls pane, the Readings sub-pane can be selected. On this pane all read-only registers on the chosen module/system is read out.



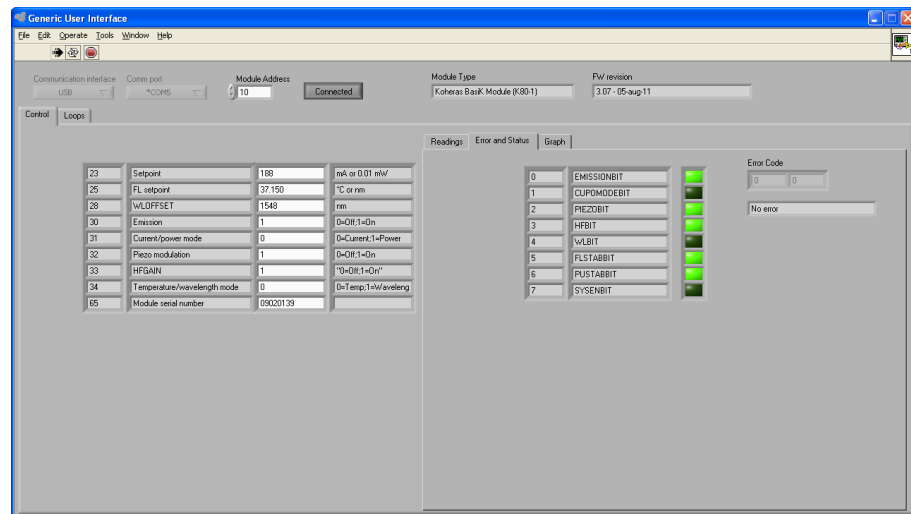
Like for the Control registers, the Reading array contains four columns. The first column shows the hexadecimal number for the register addresses, the second column a short description of the register, the third column the reading from the module/system and the fourth and last column the unit for the values.

Column 1, 2 and 4 is taken directly from the Register File and column 3 are real values read from the module/system.

Like for the Control registers, the Generic User Interface is handling scaling of the registers.

8.3.5 Error and Status

On the Controls pane, the Error and Status sub-pane can be selected. On this pane the status bits and error code from the module can be read.



Status Bits

The Status Bits is shown in an array with three columns. The first column provides the index number for the different Status Bits, the second column a short description of the Status Bits and the last column an indicator on whether the bit is high or low. A dark green color indicates a low bit whereas a light-green color indicates a logical high bit.

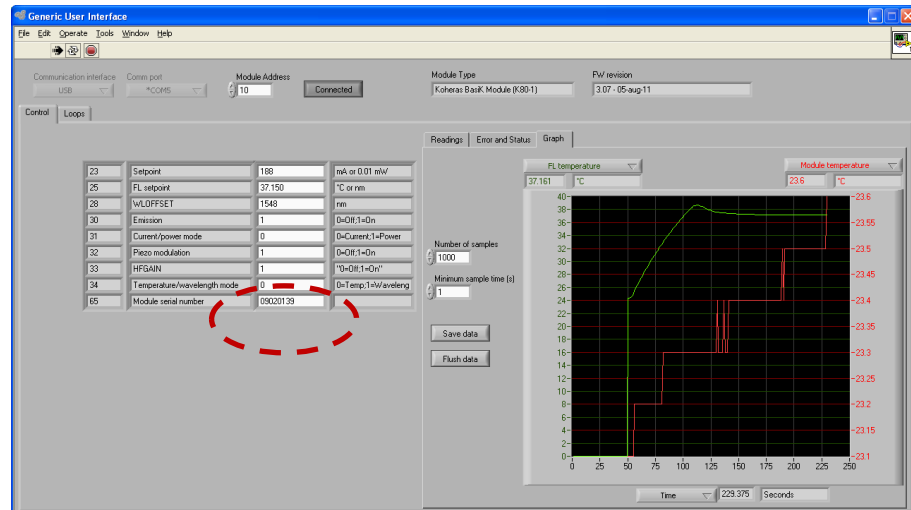
Error Code

If a module has shut down with an Error Code, this can be read out in the Error Code field. The Error Code is an 8-bit number, which is translated into a short description in the text field right underneath the Error Code field.

Description of Status Bits and Error Codes are taken from the Register File.

8.3.6 Graph

On the Controls pane, the Graph sub-pane can be selected. On this pane the content from both the Readings array and the Control array can be viewed.



X-axis

Underneath the graph the parameter that should be used for the X-axis is chosen. Possible parameters are Time, Loop Time, All Readings and All Controls parameters.

Is Time chosen the graph will show a like an oscilloscope with the cursor starting on the left side moving towards the right.

Loop Time is the time counter used for the loop sequences. See [Loops](#) for details.

Readings and Controls parameters depend of course on what type of module/system the Generic User Interface is communicating with.

Next to the X-axis selector the last value is read together with the unit.

Left-hand Y-axis

On top of the graph to the left, the parameter for the left-hand Y-axis is chosen. Exactly the same parameters can be chosen as for the X-axis.

The left-hand Y-axis is written with green text as the gridline for this is green as well as the corresponding curve in the graph.

Right-hand Y-axis

On top of the graph to the right, the parameter for the right-hand Y-axis is chosen. The same parameters as for the X-axis and the left-hand Y-axis can be chosen for the right-hand Y-axis.

The right-hand Y-axis is written with red text, as the corresponding curve and gridline is red.

Views

By using the 3 axis-selectors, it is possible to get not only oscilloscope type of views with the X-axis as a time axis, but also e.g. I-P curves (dependant on modules/systems).

Number of Samples

In this field it is chosen the maximum data points that should be shown in the graph. Right after the Generic User Interface has connected to a module/system there will be no data points to show. As data points are gathered from a module and if the number of data points are about to exceed the Number of Samples the oldest data points will be erased from the computers memory.

Minimum Sample Time

The Minimum Sample Time determines the minimum time between register values are to be updated. It is as it says a minimum value, so the actual time will be slightly

larger. Setting the Minimum Sample Time to 0, will make the Generic User Interface communicate as fast as possible.

Save Data

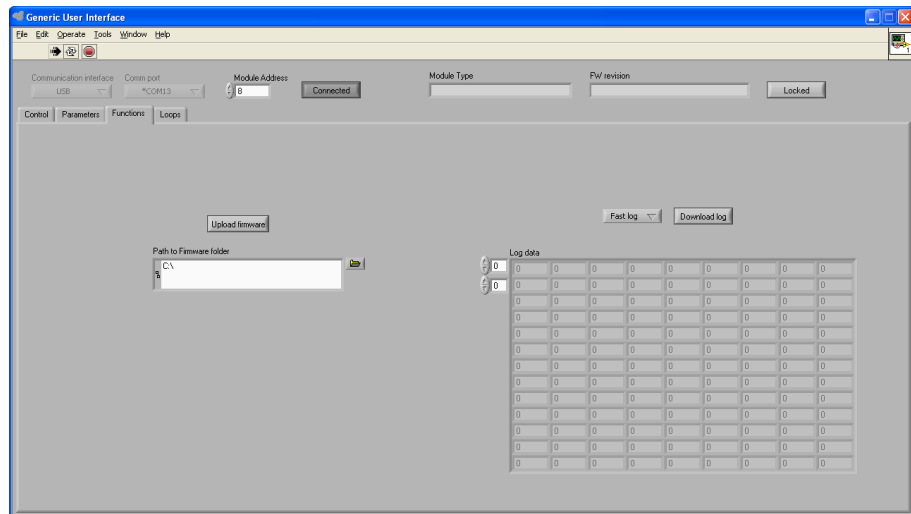
The Save Data button makes it possible to save all recorded data to a tab-delimited ASCII-file. All recorded data means not only what is shown on the graph, but also all other parameters that is not chosen.

Flush Data

Clicking on the Flush Data button will erase all recorded data, i.e. the graph will start all over again.

8.3.7 Functions

For some types of modules/systems it is possible to upload new firmware and/or download log data from the module. If these functions are possible for the module/system the Generic User Interface is communicating with, these will be available on the Functions pane.



Upload Firmware

To upload new firmware to the module the Generic User Interface is connected to, browse for the directory where the desired hex-file to be uploaded is located. Here after click on the Upload Firmware button for the upload to begin. As long as the upload is in progress the button will say Uploading Firmware.

Download Log

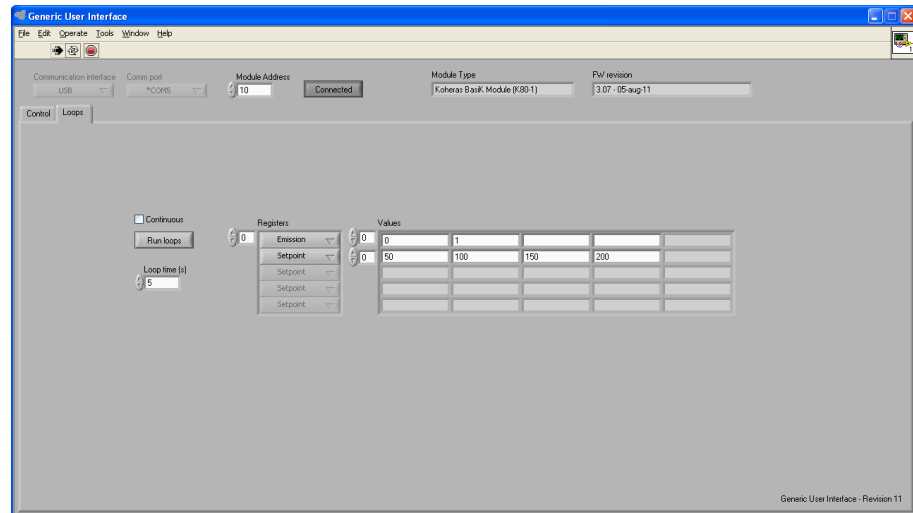
If the module the Generic User Interface is connected to features a module log-function, the log file(s) can be downloaded with the Generic User Interface. If more than one log-file is available the log-file to be downloaded is selected with the selector to the left. Click on the Download Log button to begin the actual download. The log-files can easily become very large, so it can be very time consuming to download these log-files.

8.3.8 Loops

Loops

On the Loops pane it is possible to let the Generic User Interface automatically run through a defined sequence.

The Loop sequence will step through all possible combinations of the chosen parameters.



Registers

In the Registers array the parameters for the sequence to step through is selected. The first defined parameter will be the one parameter that is changed everytime, and the last defined parameter will be the one that is changed the least times.

Values

In the Values array the different setpoints the chosen parameters to run through is defined. All values in the Values array are scaled according to the scaling factors in the Register lists.

It is not required that there should be an equal number of values for all parameters. Cells to the right of the last value can be left blank and they will be ignored.

Loop Time

Loop Time determines the minimum time the Generic User Interface should maintain each state in the sequence.

Continuous

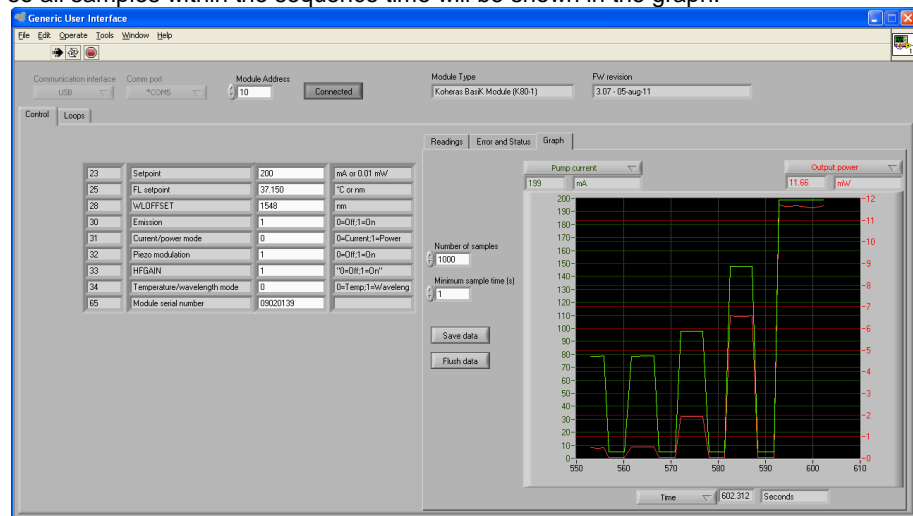
Is the Continuous check mark set, this would make the Generic User Interface run the sequence over and over again.

Run Loops

Click on the Run Loops button to start the sequence. Click on the button again to about the loop sequence if desired without disconnecting the communication between the computer and the module/system.

Graph View

On the Graph window readings from the module/system at the different states of the sequence can be viewed. Just remember to set the Minimum Sample Time to less than the Loop Time and probably also set the Number of Samples to a large number so all samples within the sequence time will be shown in the graph.



9 Appendix

9.1 LabVIEW Driver (Legacy)

Purpose

This section describes how to use and how to establish communication from LabVIEW with NKT Photonics modules/systems with digital interface, based on the NGSerialPort VI from NKT Photonics.

The functions used in this chapter is included for compatibility reasons. We recommend that new development is based on the new LabVIEW Driver API. This API is documented here: [5 LabVIEW Driver API](#).

Requirements

The module(s)/system(s) to be controlled must feature a digital interface which is connected directly or via an adaptor to the computer. The NGSerialPort VI can be utilized in LabVIEW 8.5 or any newer revision of LabVIEW.

Description

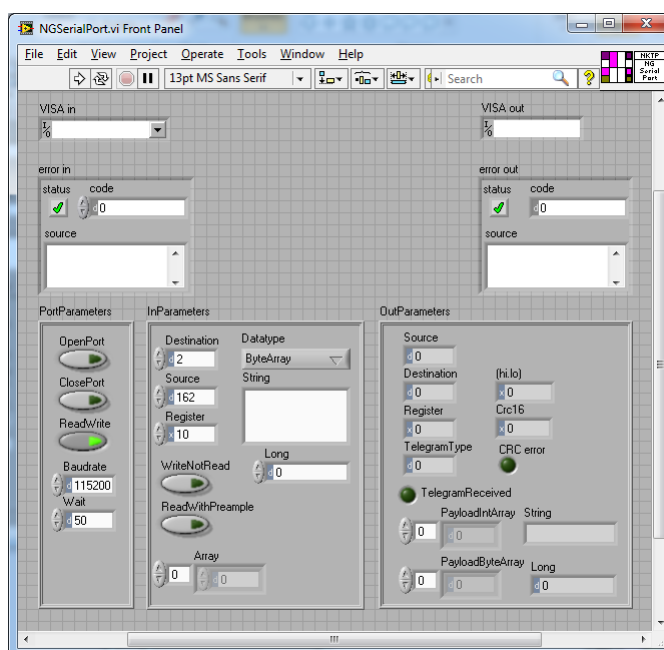
The NGSerialPort VI contains all functions needed for establishing communication from LabVIEW on the computer the NKT Photonics product(s) is connected to.

The VI is used to setup, open and close the communication port and it can be used to both read from and write to registers of different types.

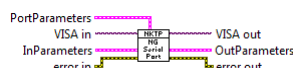
9.1.1 Inputs and Outputs

This section describes the inputs and outputs on the NGSerialPort VI, which is available both on the front panel as well as on the icon in the block diagram.

Front Panel



Icon



VISA in

This is an input, which is used to configure which COM-port to be used for communication. Please notice that even if the module(s)/system(s) may be connected to the computer with an USB connection, this connection will establish a virtual COM-port, which should be used for the communication.

VISA out

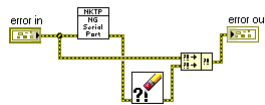
This terminal is an output, which will provide COM-port number used for the communication. This can be connected to the following NGSerialPort in the block diagram if desired.

error in

Typical error input cluster used in LabVIEW. It can be connected to the previous function with an error out. Please be aware that there will not be any communication if the error in informs about a previous error.

error out

Typical error output cluster. It can be connected to the following function with an error in. However please notice that if write commands are sent to input registers on a module/system that does not reply back with an acknowledge (ACK) or not acknowledge (NACK) the NGSerialPort VI will time out and will generate an error even though nothing is actually wrong. So in this case the error out should not be connected to the following function with an error in, but instead cleared and the input error fed out instead.

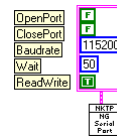


Modules and systems that do not generate ACK and NACK are:

- Koheras Basik Module (K80-1)
- Koheras AdjustiK/BoostiK System

9.1.2 PortParameters

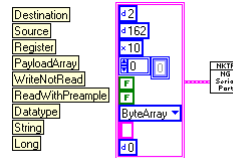
Input cluster to setup, open and close communication port and to select whether the VI should be used for communication. This input cluster can either be configured with the bundle function in LabVIEW or a constant as shown below.



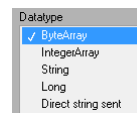
OpenPort	The communication port does not have to be opened every time LabVIEW should be communicating with the module/system. The communication port must be opened the first time (or prior) and again after the communication port has been closed if further communication is desired.
ClosePort	When no further communication is desired, the communication port should be closed with the ClosePort input. If the communication is not ended with the ClosePort command, the port will be hanging and communication cannot be established from other programs.
Baudrate	This input sets the baudrate on the computer. All NKT Photonics products communicating with the binary NKT Photonics protocol are communication with a baudrate at 115200 bits per second. The computer should have the same baudrate as the module(s)/system(s), so set this input to 115200.
Wait	This input configures how long time in milliseconds the NGSerialPort should be waiting from a response from the interfacing module/system. Typical values are 50 or 100 milliseconds.
ReadWrite	With the ReadWrite input it is selected if the NGSerialPort VI should be used for actual communication and not just for opening or closing the communication port. No matter if the VI should be used for writing or reading this input should be set high.

9.1.3 InParameters

Input cluster with information about addresses, whether the VI should be used for reading or writing, in case it should be used for writing what type of data should be written and the actual data to be written. This input cluster can either be configured with the bundle function in LabVIEW or a constant as shown below.



- Destination** This input is used to set the address of the module/system to communicate with. Module addresses are in the range from 1 to 160.
- Source** This input is used to set the address of the computer. This must be higher than 160 (A0h).
- Register** This input configures the register address to read or write.
- Datatype** In case data should be transmitted to a module/system, the Datatype selector is used to select whether it is a ByteArray, IntegerArray, String, Long or a Direct string that should be transmitted.

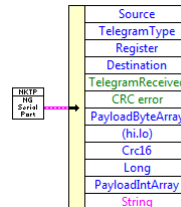


If just an integer should be transmitted, the Long function can be used.

- WriteNotRead** This input is used to configure whether the VI should be used for writing or reading. The input should be set low for reading and high for writing.
- ReadWithPreamble** Obsolete. Do not use.
- PayloadArray** If a ByteArray or an IntegerArray should be transmitted, the array should be inserted into the PayloadArray. If a single byte or integer should be transmitted, it should go into the PayloadArray by using the Build Array function leading to an array with just a single element.
- String** When a String should be transmitted it should be inserted into the String input.
- Long** Input for Long values.

9.1.4 OutParameters

Output cluster to with information about addresses, whether the VI should be used for reading or writing, in case it should be used for writing what type of data should be written and the actual data to be written. The output can be splitted up with the Unbundle By Name function as shown below.



Source

This output provides the sender address of the received telegram.

TelegramType

The TelegramType output provides a general information about the received telegram.

Code	Type	Description
0	NACK	Response. Message not understood, not applicable, or not allowed.
1	CRC error	Response. CRC error in received message.
2	Busy	Response. Cannot respond at the moment. Module too busy.
3	ACK	Response. Received message understood.
4	Read	Query. Read the contents of a register.
5	Write	Transmission. Write something in a register.
8	Datagram	Response. Register content returned; caused by a previous "Read".

Register

This output provides the register address for the received telegram.

Destination

The Destination output provides the receiver address of the received telegram. If the telegram is a response to a Read or Write request, the destination address in the response will be identical to the source address in the request, and vice versa.

TelegramReceived

Boolean output, which is low if no telegram is received and high if a telegram is received.

CRC error

The binary protocol utilizes a cyclic redundancy check (CRC) to ensure that data is not corrupted during the transmission. When the NGSerialPort VI receives a telegram, it will verify the CRC value. In case the telegram was corrupted during the transmission, it will report a CRC error (high if error).

PayloadByteArray	This output array provides the received data as bytes in array form.
(hi.lo)	Do not use. For development use only.
Crc16	The Crc16 output provides the CRC value for the received telegram.
Long	This output provides the received data as a Long value.
PayloadIntArray	This output array provides the received data as integers in array form.
String	The String output provides the received data as a String.

9.1.5 Using the NGSerialPort

This section provides a small example of how to use the NGSerialPort from the front panel. This example shows how to read out the wavelength #0 setting from a RF driver driving a SuperK Select and to write a new setpoint.

VISA in

Set the VISA in to the actual communication port. In this case it is COM10.

PortParameters

OpenPort and ClosePort are both set high in order to properly open and close the communication port. (In a final LabVIEW application calling the NGSerialPort multiple times it is not needed to open and close the communication port every time, but just make sure to open the port in the beginning and in the end of the program).

ReadWrite is set high, as it is desired to read a register value.

Baudrate is set to 115200 (default value) and Wait is set to 50 (default value).

9.1.5.1 Read Example

InParameters

In this example it is an external RF driver driving the SuperK Select. The address of the RF driver is determined by the rotary switch on the back of the driver. The small arrow in the rotary switch is pointing at no. 3. Adding 0x10 to the number on the rotary switch makes the module address 0x13. This number is used in the destination address field.

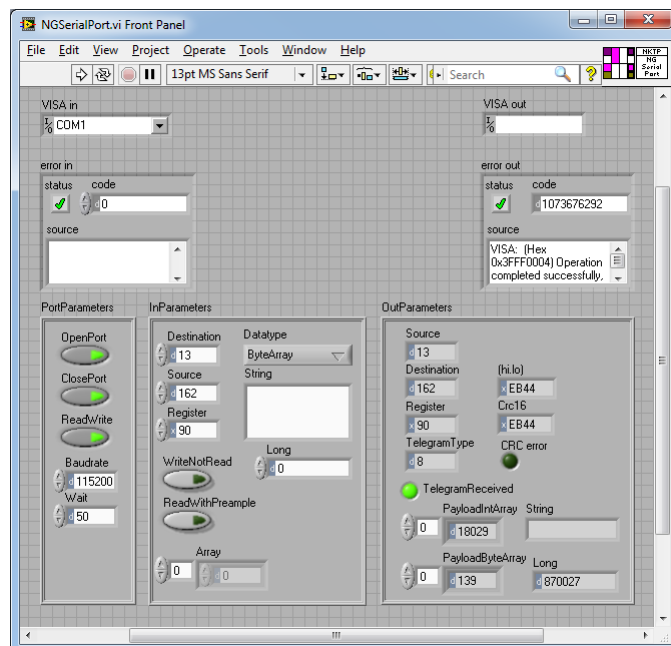
The Source address field is set to address 0x42 (default value).

Wavelength #0 is in the register file for the RF driver (66.txt) is found to be 0x90, which is typed into the Register field.

The WriteNotRead is set low to read from (and not write to) a register.

Run

Run the VI by clicking on the arrow in the top left corner of the front panel.



OutParameters

After the VI has run, the TelegramReceived indicator should indicate that a telegram has been received.

The received telegram is presented both in the PayloadIntArray, PayloadByteArray, String and Long fields. As the actual register is a 32-bit register (seen from the 66.txt register file), the wavelength is directly read out from the Long field as 870027 pm or 870.027 nm.

9.1.5.2 Write Example

InParameters

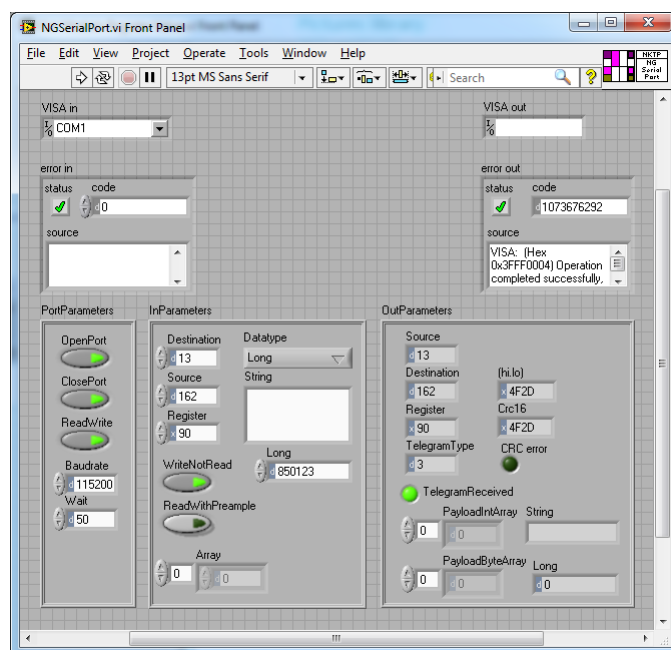
To write a new wavelength to the wavelength #0 register, change the WriteNotRead to high.

Set the Ring selector (Datatype) to Long, as a 32-bit register value should be transmitted.

Type in the new desired wavelength into the Long field (in this example 850123 for 850.123 nm).

Run

Run the VI using the arrow in the top left corner of the front panel.

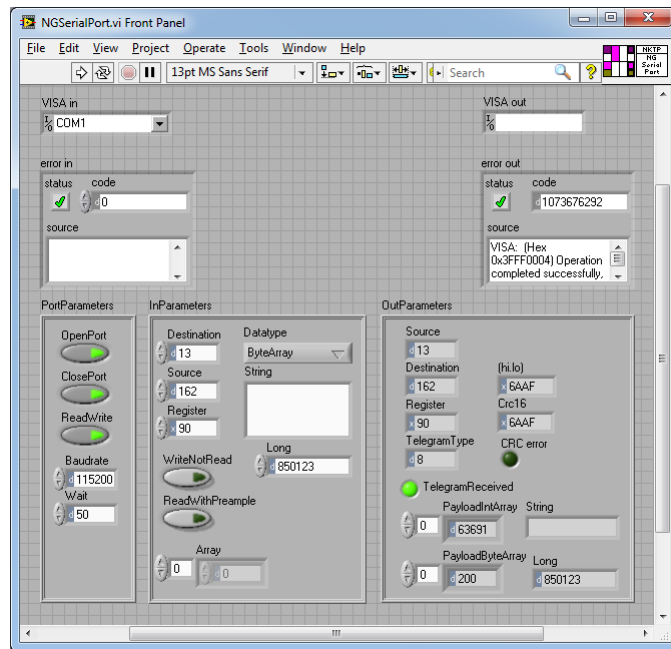


OutParameters

When the RF driver receives this command, it should reply back to the computer with an acknowledge (telegram type 3). This is seen in the TelegramType field.

Verification

To verify that the new setpoint is written and understood by the RF driver, click WriteNotRead low and remain other settings unchanged to read out the new setpoint and run the VI again.



In the Long field in the OutParameters the new setpoint is seen.

9.1.5.3 Writing other types of registers

- Byte(s)** If you want to transmit a byte or a byte array, set the Ring selector (Datatype) to ByteArray. For a single byte only use the first cell in the Array.
- Integer(s)** If a 16-bit integer or integer array should be transmitted, set the Ring selector to IntegerArray. Type in a single integer into the first cell in the Array, or use following cells in the Array for multiple integers.
- String** If a string should be written to a module, set the Ring selector to String and write the string into the String field.
- Long(s)** A single 32-bit long value can be written to a module as shown in the example above. If a 32-bit long array should be written to a module, it should be split into a 16-bit array with the least significant 16-bit value first followed by the most significant value (illustrated in the figure below).

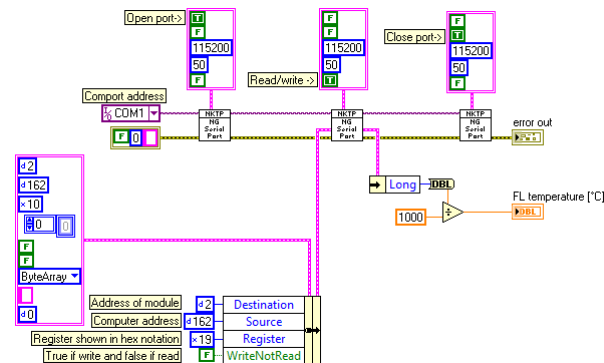
Long#0		Long#1		Long#2		...
Int#0(Least Significant)	Int#0(Most Significant)	Int#1(Least Significant)	Int#1(Most Significant)	Int#2(Least Significant)	Int#2(Most Significant)	...

9.1.6 Programming Examples

This section provides two programming examples, i.e. an example for reading a value from a module and another example for writing a value to a module.

9.1.6.1 Reading Example

Following an example is shown for how to read out a value from a module.



Opening Port

First time the NGSerialPort VI is called, it is only for opening the communication port. In this example COM1 is used for the communication port.

Building Up Telegram

Before the NGSerialPort VI is called for the second time, the InParameters cluster is configured with a constant and Bundle By Name function. In this example the module to communicate with has address 2, the computer is given address 162, and the register to read is 19h.

In the block diagram, a "d" in front of a value signifies decimal, and an "x" signifies hexadecimal. This feature is activated by right-clicking on a numeric control, and selecting Visible Items / Radix.

Requesting and Receiving Telegram

When the NGSerialPort VI is called the second, time the PortParameters are set to read/write.

After the NGSerialPort VI has been called the second time, the type of received data is picked out with the Unbundle By Name function, and in this case scaled with a factor of 1000 before it is shown in the FL temperature indicator.

Closing Port

Last before ending the program the port is closed by setting the second boolean value high in the PortParameter cluster for the third call of the NGSerialPort VI.

9.1.6.2 Example files

In addition to the LabVIEW driver, some LabVIEW example files are included in the Software Development Kit.

From ByteArray.vi	Converts a byte array to an array of a different type (single-precision float, 16-bit unsigned integer or 32-bit unsigned integer). Necessary byte swapping is built in. It is a polymorphic function, that contains <i>From ByteArray([SGL]).vi</i> , <i>From ByteArray([U16]).vi</i> and <i>From ByteArray([U32]).vi</i> instances.
To ByteArray.vi	Converts an array of single-precision floats, 16-bit integers or 32-bit integers to a byte array. Necessary byte swapping is built in. It is a polymorphic function, that contains <i>To ByteArray([SGL]).vi</i> , <i>To ByteArray([U16]).vi</i> and <i>To ByteArray([U32]).vi</i> instances.
Get Module Firmware Version.vi	Reads the firmware version code in the module on the specified address.
Get Module Register Value.vi	Opens a serial port, reads a numeric value from a register, and closes the serial port.
Set Module Register Value.vi	Opens a serial port, writes a numeric value to a register, and closes the serial port.

NKT Photonics

Blokken 84

3460 Birkerød

Denmark

Phone: +4543483900

Fax: +4543483901

www.nktphotonics.com