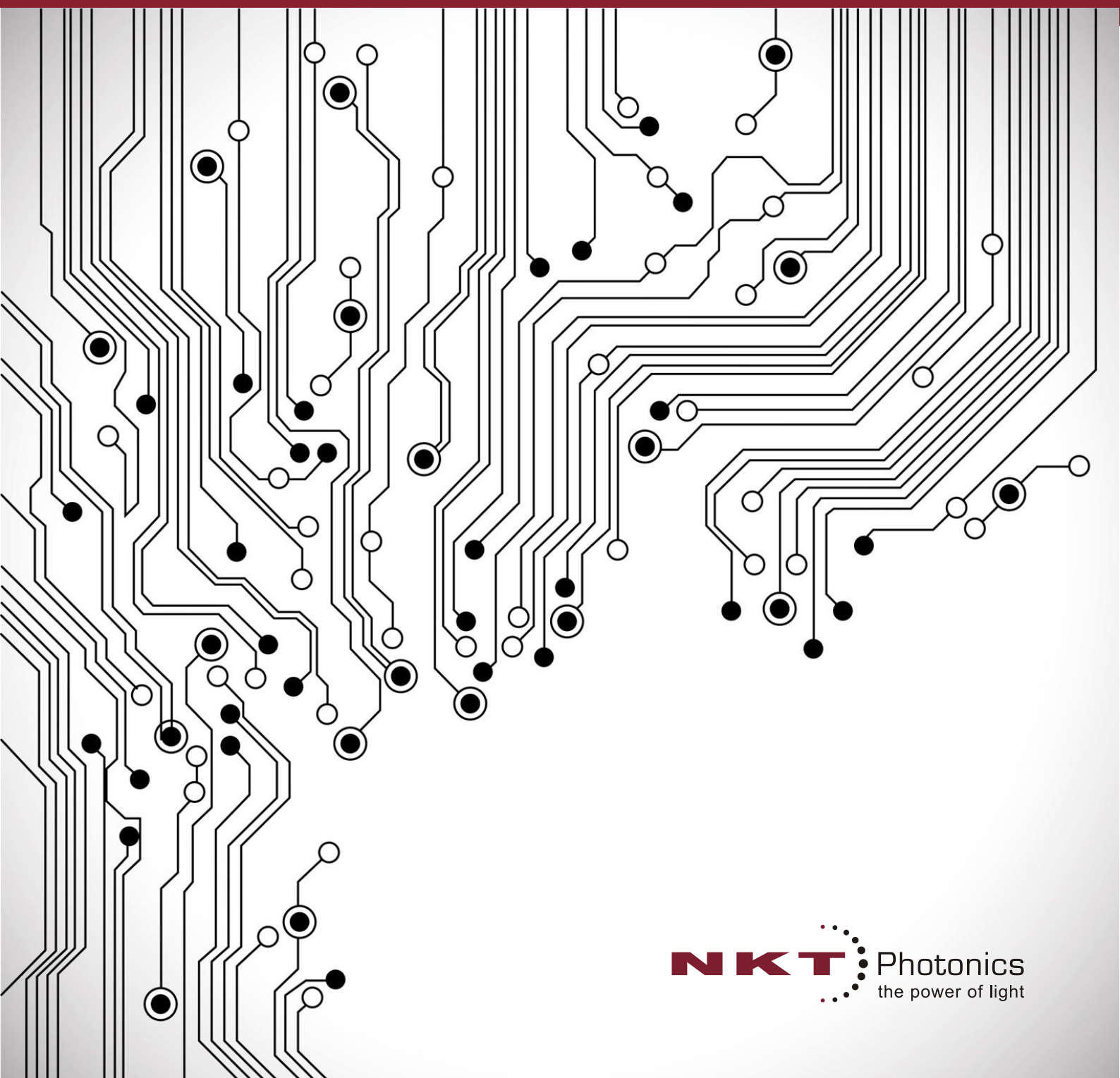


SDK

Software Development Kit for NKT Photonics Instruments

Instruction Manual



Version: 2.1.7
Published: Nov 2021
Author: HCH, TOO, ETH, MKU, TVA, AKL, ANRE
Copyright © 2019 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	Example files	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	Registers	30
6.1	Register files.....	30
6.2	General registers	33
6.3	Koheras AdjustiK/BoostiK System (K81-1 to K83-1)	34
6.4	Koheras ADJUSTIK/ACOUSTIK System (K822 / K852).....	35
6.4.1	General settings.....	35
6.4.2	Readouts	35
6.4.3	Modulation, master module.....	36
6.4.4	Modulation, laser modules	37
6.4.5	Ethernet	37
6.4.6	Special options.....	37
6.5	Koheras BasiK Module (K80-1)	38
6.6	Koheras BASIK MIKRO Module (K0x2).....	40
6.6.1	General settings.....	40
6.6.2	Readouts	41
6.7	Koheras BASIK Module (K1x2)	42
6.7.1	General settings.....	42
6.7.2	Readouts	43
6.7.3	Modulation	44
6.8	BoostiK OEM Amplifier (N83)	45
6.9	KOHERAS BOOSTIK Line Card (K2x2x)	46
6.10	SuperK EXTREME System (S4x2) and SuperK Fianium	47
6.10.1	Main module	47
6.10.2	Front panel.....	48
6.11	RF Driver (A901) for SuperK SELECT.....	50
6.12	SuperK SELECT (A203).....	52
6.13	SuperK VARIA (A301).....	53
6.14	Extend UV (A351).....	54
6.15	SuperK COMPACT (S024).....	55
6.16	aeroPULSE (P000).....	58
6.16.1	Control Unit.....	58
6.17	SuperK EVO (S1xx, S2xx, S3xx)	60
6.17.1	Main module	60
6.17.2	Ethernet module.....	62
6.18	SuperK FIANIUM (S4x3)	63
6.18.1	Main module	63
6.18.2	Ethernet module.....	64
6.19	aeroPULSE G3 (P4xx, NP4xx)	66
6.19.1	Main module	66
6.19.2	Femtoplane Pulse Picker	70
7	Silabs USB Driver.....	71
8	Generic User Interface	73
8.1	Installing the software	73
8.2	Register File Path	76
8.3	Using the Generic User Interface Software	76
8.3.1	Front Panel	76
8.3.2	Starting Up.....	78

8.3.3	Controls	79
8.3.4	Readings.....	80
8.3.5	Error and Status.....	81
8.3.6	Graph.....	82
8.3.7	Functions	83
8.3.8	Loops	84
9	Appendix.....	86
9.1	LabVIEW Driver (Legacy)	86
9.1.1	Inputs and Outputs.....	86
9.1.2	PortParameters.....	88
9.1.3	InParameters	89
9.1.4	OutParameters.....	90
9.1.5	Using the NGSerialPort.....	92
9.1.6	Programming Examples.....	95

1 Introduction

Notice

Before using NKT Photonics' Software Development Kit (SDK), it is recommended to read this document. It contains important information about how to get started with the SDK and the details of its architecture, tools and implementation examples.

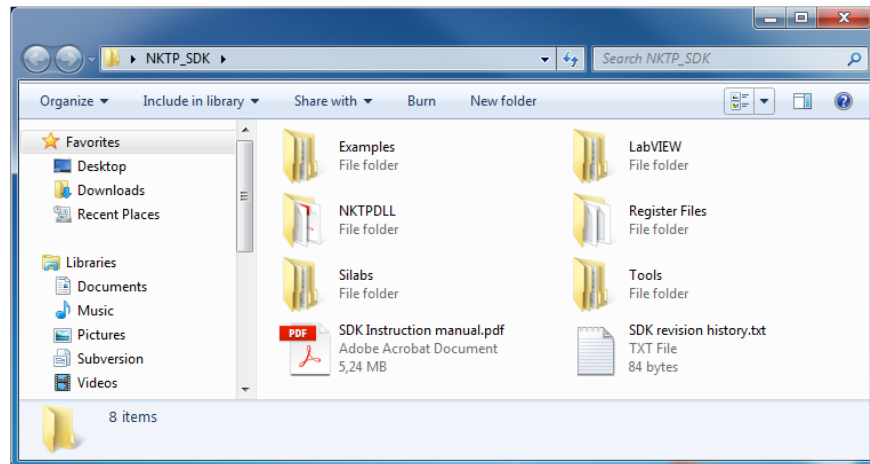
Warning

Laser products are potentially dangerous. Ensure to read the instruction manual for the laser, amplifier and/or accessory before operating it.



Contents

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



Manual

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

Protocol Description

The [Interbus Protocol Description](#) section is a low-level description to help you write your own code. You may for example, require code for a micro controller in your own system that can communicate with NKT Photonics modules/systems.

NKTPDLL

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

Examples

The SDK includes multiple code examples, the [Examples](#) section explains how to use them.

LabVIEW Driver

The [LabVIEW Driver](#) section describes how to use the included LabVIEW driver along with the NKTP LabVIEW API installed in the *LabVIEW\Labview Driver* folder.

Register Files

The [Register Files](#) section describes how to read and use information from the register files located in the *Register Files* folder. Each NKTP module or system requires a separate register file for communication.

Silabs USB Driver

Silabs USB driver is required for communication with NKT Photonics products that include a USB interface such as for example, SuperK FIANIUM and Koheras ADJUSTIK laser. The driver is located in the *Silabs* folder and the [Silabs USB Driver](#) section describes how to install it.

Generic User Interface

The Software Development Kit includes a Generic User Interface software tool. The tool uses information from register files in the *Register File* folder of the SDK folder hierarchy. When developing and testing your own code, the tool is useful to help understand the registers and monitor code that writes to and reads from the device 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. **NOTE:** This protocol is not INTERBUS, an industrial communication technology developed by Phoenix Contact.

2.1 Physical

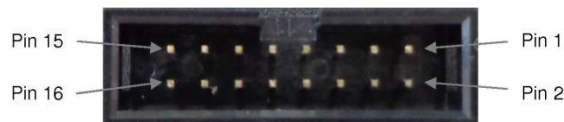
Physical

The NKT Photonics Interbus standard is based on 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 (see 2.2 Telegram) carried are built in the exact same way.

RS-485, modules

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

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

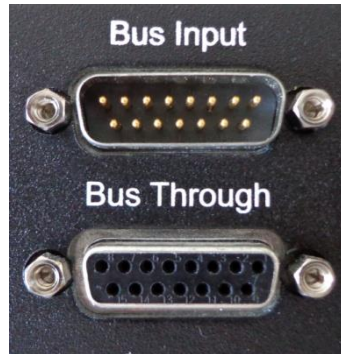


The IDC connector pinout

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 forward 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 return 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 refer to the individual module’s user manual.

External accessory modules are not equipped with an 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 (i.e. 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 forward Interlock loop signal input (5 V). This input is 12 V tolerant in most modules.
4	O	Interlock loop return 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 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

NOTE: In female connectors, the signal directions are opposite.

RS-485, master

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

The bus output can supply multiple accessories with 12 V supply voltage. The absolute maximum current consumption is 4 A. In some products, the output is protected with a fuse, which cannot be replaced by customers. **CAUTION:** 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 forward 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 return 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. Ensure to keep these signals isolated from other equipment, either by mechanical switches or by relays.

When the external bus is not used, ensure to connect the bus defeater to the port, 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. Do not attempt to communicate directly through this port. Instead, use either an available RS-232, USB or Ethernet port. If necessary, telegrams with an external destination are relayed to the external bus by the laser system.

RS-232

Some systems are equipped with a standard RS-232 interface, using 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 the None mode is set, 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 device 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

In addition to the USB / RS-232 serial ports, some products are also equipped with an Ethernet port using TCP over IP. The TCP port number is by default, set to 10001 and a static IP address is initially configured over a serial port connection. Depending on the product, DHCP may also be available for address assignment. Check the register lists for further details.

NOTE: A few legacy products, however, use UDP over IP and have a fixed IP address. The address is usually printed on a label attached to the product. These products are not covered by this manual.

NOTE: The transmitted and received telegrams through Ethernet are identical to those sent using serial communication. 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
------	--------	------	-------	-------------------	---------	---------

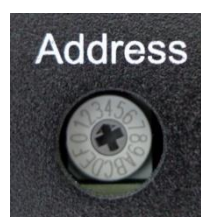
Destination address

The destination address specifies which module the message is intended for. Each module on a bus has its own unique address, which is a number between 1 and 160.

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

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

Some external modules, such as laser accessories, are designed to share an external bus with several other external modules. These modules have a bus address selector, which is a rotary switch with 16 positions. You can use the switch to set the external module's address. The actual address assigned is not 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 in the center of the switch points to the address setting (0..F). However, the actual address is calculated by adding 16 (0x10) to the setting. In the picture above, the setting is "3", meaning that the actual address is 19 (0x13).

NOTE: When communicating with a single laser system, the system may internally contain several modules, each with its own address. Therefore, a 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 indicates which module has sent the message, thus where a response should be directed. For host equipment (such as a computer), the source

address must be greater than 160 (48 in legacy products). If you are creating new interface software, use a host address above 160 to avoid future address conflicts.

Message type

The message type (see table below) indicates 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 sets the corresponding bit in the register value. Logic zeros have no effect.
7	Write CLR1	Transmission. Writing a logic one to a bit clears 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 inverts (toggles) 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 responds with a type 8 (datagram) if a type 4 read message was sent, or 3 (acknowledge) if a type 5, 6, 7 or 9 write message type was sent.

Register number

The register number indicates to the specific module what exactly the message is about. It consists of a single byte. Each module has its own list of predefined 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 a host tries to access a non-existing or restricted register, a module answers 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 can vary considerably, but multi-byte values (more than one byte in size) should be transmitted little-endian, i.e. LSB first.

Byte 0	Byte 1
16-bit value	
Bits 0-7	Bits 8-15

Byte 0	Byte 1	Byte 2	Byte 3
32-bit value (integer or float)			
Bits 0-7	Bits 8-15	Bits 16-23	Bits 24-31

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

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
64-bit value							
Bits 0-7	Bits 8-15	Bits 16-23	Bits 24-31	Bits 32-39	Bits 40-47	Bits 48-55	Bits 56-63

CRC

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

Byte n	Byte n+1	Byte n+2
CRC value		
	Bits 8-15	Bits 0-7

Prior to calculation, the CRC value must be set to 0. Below is an example on how to implement a CRC algorithm in C.

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

uint8_t data_byte;
static uint16_t crc_value;

crc_value = (uint8_t)(crc_value >> 8) | (crc_value << 8);
crc_value ^= data_byte;
crc_value ^= (uint8_t)(crc_value & 0xff) >> 4;
crc_value ^= (crc_value << 8) << 4;
crc_value ^= ((crc_value & 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 `data_byte`, and the CRC result ends up in the variable `crc_value`.

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).

"0FA20466" (hex)	
1 byte checksum	27
CRC-16	0x1C20
CRC-16 (Modbus)	0x3820
CRC-16 (Sick)	0x9BFE
CRC-CCITT (XModem)	0xC7B6
CRC-CCITT (0xFFFF)	0x4376
CRC-CCITT (0x1D0F)	0xC9A6
CRC-CCITT (Kermit)	0xC669
CRC-DNP	0x19A2
CRC-32	0x6366472D

0F A2 04 66

Input type: ☐ ASCII ☒ Hex

NOTE: 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 the transmission routine.

Address cycling

In multi-tasking systems, it can be difficult to associate the received responses that correspond with the original transmitted telegrams. Although NKT Photonics modules and 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 161 (0xA1) to 255 (0xFF) can be used for this.

Address scan

It is possible to invoke an address scan to check for modules available at each 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 an address, it responds with its module type number to identify itself. If no response is received before the timeout expires, no module is present on that address.

Arrays

In some cases, the data bytes contain an array of separate values. In the register files and in this manual, these arrays are treated as zero-based. This means that the first element in the array starts at 0 (the 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 transmitting a telegram, any special characters within the message framing bytes 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 other than Start of Telegram and End of Telegram. This ensures that the telegram framing is discernable for all Interbus modules listening.

CRC byte conversion

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.

Example

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 (uint8_t 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 emission control 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 emission on")
BC	CRC most significant byte
E1	CRC least significant byte

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 an “Acknowledge” response is activated (by default turned off in a K80-1 BasiK module). The module transmits an “Acknowledge” message as follows (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), the register is 0x11, and the temperature measured is 37.214 °C or 37214 m°C (millidegrees).

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 fiber laser temperature)
75	CRC most significant byte
83	CRC least significant byte

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]

The response from the BasiK module includes the measured temperature value mentioned above and looks like this (special character substitutes are highlighted):

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

With framing removed:

[A2][0A][08][11][5E][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
uint16_t CRC_add1(char data, uint16_t crc)
{
    crc = (uint8_t)(crc >> 8) | (crc << 8);
    crc ^= data;
    crc ^= (uint8_t)(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
uint16_t tx1(char data, uint16_t 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)
{
    uint16_t 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 the 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 also be implemented, such as catching UART overrun errors etc.

```

_Bool TelegramReady, InFrame, SpecialChar; // boolean
char RxCounter, RxBuffer[BUFLNGTH];
uint16_t 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;
                    SpecialChar = 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 64-bit versions.

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

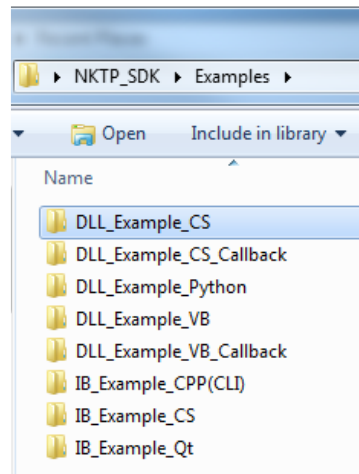
The API also contains a set of functions supporting advanced callback features, where the DLL kernel is constantly monitoring all the connected devices and their registers, generating callbacks to usercode when things change.

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

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

4 Example files

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.

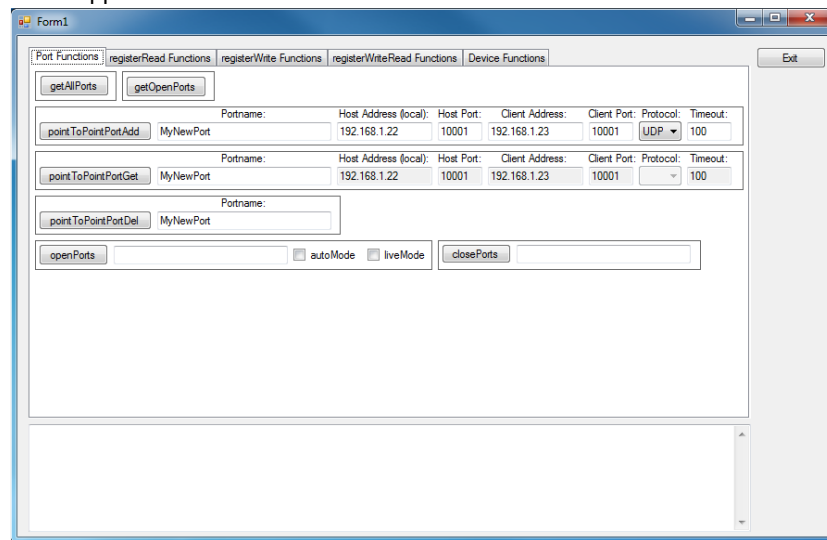
Requirements

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.

Copy this file to your project folder and include the file in your project and then 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 the first time you load the NKTPDLL in the Visual Studio debugging mode. You could disable this warning by going to Debug-Exceptions-Managed Debugging Assistants and turning off 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.

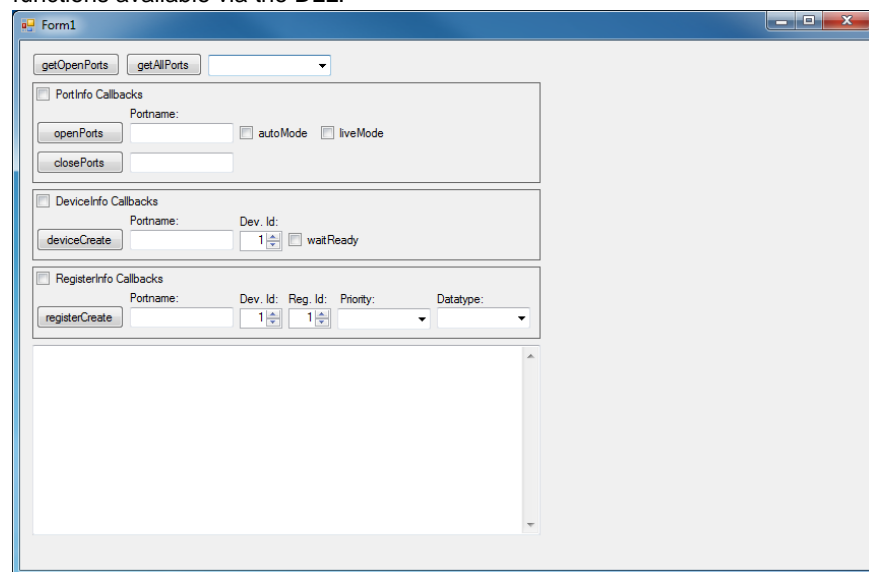
The Visual Studio Community(express) edition could be downloaded from Microsoft's website.

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.

Copy this file to your project folder and include the file in your project and then 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.

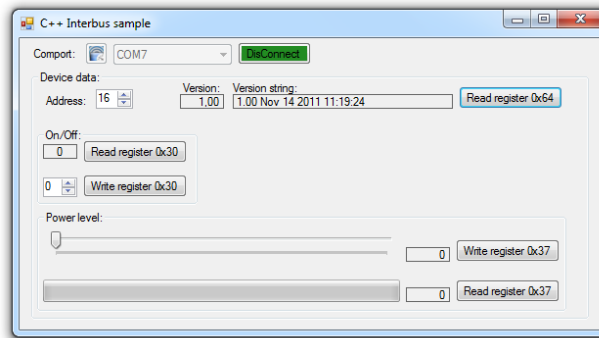
NOTE: You might get the "LoaderLock" exception first time loading the NKTPDLL in the Visual Studio debugging mode. You could disable this warning by going to Debug-Exceptions-Managed Debugging Assistants and turning of the LoaderLock warning.

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 use 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) or 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>The following is a very basic and simple example using the NKTP_DLL.py script to turn on emission on a SuperK 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

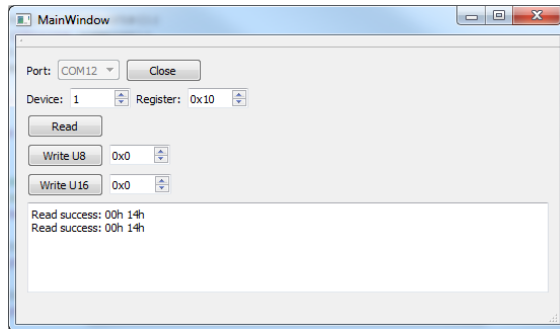
- Purpose** This section describes the CPP(CLI) and the C# example application.
- Requirements** The module(s)/system(s) to be controlled must feature a digital interface which is connected directly or via an adaptor to the computer. To compile and run the example code, you will need a Visual Studio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



NOTE: The sample code does not implement retransmission on CRC error in the communication, 3 to 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.



NOTE: The sample code handles retransmission on CRC error in contrast to the C++/CLI sample which does not.

- Files** The sample is a complete self-running sample, using the Qt framework. The folder *Interbus* contains an *Interbus.pri* file which is easily included in your own project. Copy the *Interbus* folder to your own project and include the *Interbus.pri* file in your own project file and include the *IBHandler.h* file where you need Interbus functionality.

The *IBHandler* class contains a set of functions used to handle Interbus communication. 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 *mainwindow.cpp*

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.

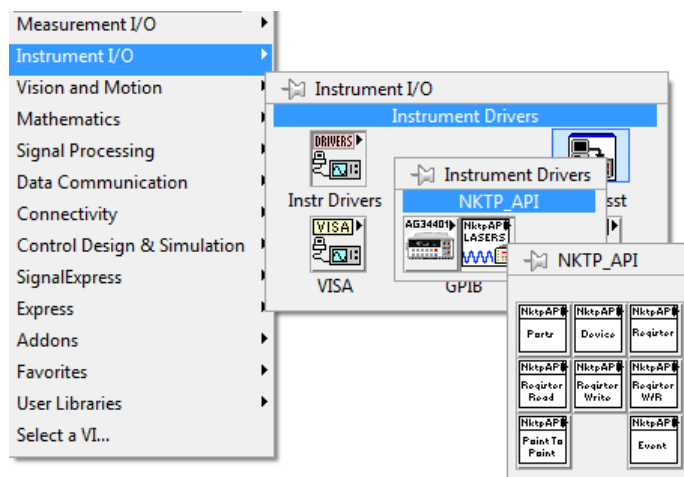
5.1 Adding the NKTP LabVIEW VIs to the LabVIEW palettes.

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 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 can 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 a VI template included that can be used if you want to register LabVIEW User Events to asynchronous events from the NKTP.dll driver. 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 the following structure.

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
ReqWVR
(U8)

NktpAPI
ReqWVR
(S8)

NktpAPI
ReqWVR
(U16)

NktpAPI
ReqWVR
(S16)

NktpAPI
ReqWVR
(U32)

NktpAPI
ReqWVR
(S32)

NktpAPI
ReqWVR
(F32)

NktpAPI
ReqWVR
(U64)

NktpAPI
ReqWVR
(S64)

NktpAPI
ReqWVR
(F64)

NktpAPI
ReqWVR
(Arcii)

NktpAPI
ReqWVR
(Void)

NktpAPI
ReqWVR
(Void)

NktpAPI
ReqWVR
(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 Registers

6.1 Register files

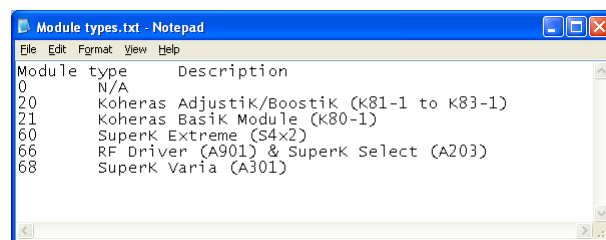
Each product or module has a specific set of registers for communication. These registers are listed in the sections below, and in the register files that are part of the Software Development Kit. This section handles the register files, which are simple text files, with a specific format.

Notice

In systems like for example, 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 for example Microsoft Notepad to get an overview of the different types of modules.



File naming

The module types file shows, that a SuperK EXTREME system has the module type 60h. This means, that the register file for the SuperK system is named 60.txt.

How to read register files

Open the desired register file in for example, Microsoft Excel. The register files are tab delimited ASCII files, which Microsoft Excel can open and display.

As the files are text files, they can also be opened in Notepad and other suitable text editors.

Register Address	Register Description	Unit	Type
Module type			
SuperK Extreme (S4x2)			
Readings			
11	NTC1 temperature	°C	I16
Controls			
30	Emission	0=Off;3=On	U8
31	Setup bits	0=Current mode;1=Power mode	U16
32	Interlock	(>0=reset interlock)	U16
34	Pulse-Pickerratio	Times	U16
35	Pulse-Pickerratio	ns	U8
36	Watchdog interval	Seconds	U8
37	Power level	%	U16
38	Current level	%	U16
65	Module serial number		string
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		

Readings and controls

The section under Readings includes read only registers. The Controls section includes the 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

The first column contains the register numbers under their respective sections. For example, reading register 11h from module type 60h will return a measured temperature.

Register description

In the second column there is a brief description of the registers.

Unit

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

Type

The fourth column provides register data types. Remember to distinguish between signed and unsigned datatypes.

Type	Variable	Range
U8	Unsigned 8-bit Integer	0..255
U16	Unsigned 16-bit Integer	0..65535
U32	Unsigned 32-bit Integer	0..4294967295
I8	Signed 8-bit Integer	-128..127
I16	Signed 16-bit Integer	-32768..32767
I32	Signed 32-bit Integer	-2147483648..2147483647
H8	Hexadecimal 8-bit Integer	00h..FFh
H16	Hexadecimal 16-bit Integer	0000h..FFFFh
H32	Hexadecimal 32-bit Integer	00000000h..FFFFFFFFh
string	Text string	

NOTE: The hexadecimal datatypes do not distinguish between signed or unsigned, but rather display values as purely hexadecimal.

Scaling

The last column specifies a scaling factor, that converts the integer value to the actual value that fits the unit.

Example 1: The value 287 is read from an "Inlet temperature" register, with the unit °C, and datatype I16. The scaling factor for this register is 0.1. The register value 287 multiplied with 0.1 gives 28.7°C.

Example 2: The same register now returns -5 (translated as signed value from hexadecimal FFBh). Multiplied by 0.1, this gives -0.5°C.

6.2 General registers

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

Module type (61h)

Register address 61h indicates which module the host computer is communicating with. This is used to establish or verify the type of module the host computer is connected to.

In legacy modules, this is an 8-bit value. In later modules, this has been expanded to 16 bits.

NOTE: A few specific modules return two bytes, but only the first byte is the module type. These modules are:

K81-1, K82-1 and K83-1 (module type 20h)

K80-1 (module type 21h)

Serial number (65h)

From 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)

From register address 66h the status of a module/system can be monitored. The status bits are combined in either one or more bytes. For an explanation of the different status bits, refer to the respective Register file.

Error code (67h)

From 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. Refer to the respective Register file for more information about the various error codes.

NOTE: *Koheras AdjustiK/BoostiK systems (K81-1 to K83-1) and Koheras BasiK module (K80-1) do not generate error codes.*

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

This section describes specific registers of Koheras AdjustiK/BoostiK systems that have product numbers starting with K81-1, K82-1 and K83-1.

Module type and module address	For Koheras AdjustiK/BoostiK systems, 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 these systems. The actual output power level and estimated wavelength cannot be monitored through the Interbus interface.
Setpoint (23h)	Depending 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 units (10 µW).
Fiber laser setpoint (25h)	<p>Depending 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 fiber laser setpoint register. The fiber laser setpoint value is an unsigned 16-bit integer value. Temperature is set in milli-degrees C (1/1000 °C) and the wavelength setting is made in pm and is relative to the wavelength offset value.</p> $\text{Total wavelength [nm]} = \text{Wavelength offset [nm]} + \text{Fiber laser setpoint [pm]} / 1000$
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 fiber laser 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.

6.4 Koheras ADJUSTIK/ACOUSTIK System (K822 / K852)

6.4.1 General settings

This section describes specific registers of Koheras ADJUSTIK/ACOUSTIK systems that have 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 protocol	TCP
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. Refer to the laser module register list in section 6.5.
Interlock (32h)	If the door interlock is connected and in the closed state, the key switch on the front plate is in On position and the External bus interlock circuit is looped back with for example, a bus defeater terminating the bus, 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 automatically shut-off (laser emission only – not electrical power) in case communications is lost with the host. The value in the watchdog register determines how many seconds without communication the system tolerates before disabling emission. If the register value is set to 0, the feature is disabled – 8-bit unsigned integer. Maximum value is 255 seconds.
Broadcast wavelength offset (2Dh)	Values written to this register are broadcast to the wavelength offset register in all laser modules.
Broadcast power, dBm (2Eh)	Values written to this register are broadcast to the dBm power setpoint register in all laser modules.
Broadcast power, mW (2Fh)	Values written to this register are broadcast to the linear (mW) power setpoint register in all laser modules.

6.4.2 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.

6.4.3 Modulation, master module

Wavelength modulation frequency (22h) This register sets 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 in 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 in tenths of a 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 in tenths of a percent (permille, ‰) in a 16-bit signed integer.

Amplitude modulation frequency (26h) This register sets 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 in 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 in tenths of a percent (permille, ‰) in a 16-bit unsigned integer.

Amplitude modulation depth (29h) Amplitude modulation depth. The resolution is in tenths of a percent (permille, ‰) in a 16-bit unsigned integer.

Modulation setup (3Bh) This register is configured the modulation setup as a 16-bit integer.

Bit 0: Amplitude modulation frequency selector.

Bit 1: Not used

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: Not used

Bit 8: Wavelength modulation frequency selector.

Bit 9: Not used

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: Not used

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

6.4.4 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. The register format is an 8-bit integer.</p> <p>This automatically enables or disables 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. The register format is an 8-bit integer.</p> <p>This automatically enables or disables the external modulation source in the laser modules.</p>

6.4.5 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. For example, 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 in a 16-bit integer.</p>
Host port (B5h)	<p>If this value is zero, the module responds to any sender. Otherwise, it only responds to the port number set in this register as a 16-bit integer.</p>
Host IP address (B7h)	<p>Host (computer) IP address. If this value consists of zeroes only, the module responds to any sender. Otherwise, it only responds to telegrams from a host with the IP address set in this register.</p>

6.4.6 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 create software for multichannel systems equipped with 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 using address 1 is “cloned”, to make the control software believe that there is more than one laser module.</p>
--------------------------------------	---

6.5 Koheras BasiK Module (K80-1)

This section describes specific registers for Koheras BasiK modules that have 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 turns off laser emission from the Koheras BasiK Module. Writing 1 to this register turns on laser emission if the interlock circuit is closed.
Current/power mode (31h)	If the laser module can operate in either current or power mode, it is possible to shift between the two modes with the current/power mode register. The value 0 sets the laser module to operate in current mode. The value 1 sets the laser module to operate in power mode.
Setpoint (23h)	Depending 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 milliamps and output power is set in hundredths of a milliwatt to give a resolution of 10 microwatts (μ W).
Fiber laser setpoint (25h)	Depending 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 fiber laser setpoint register. The fiber laser setpoint value is an unsigned 16-bit integer value. Temperature is set in milli-degrees C. The wavelength setting is made in picometers (pm) and is relative to the wavelength offset value. $\text{Total wavelength [nm]} = \text{Wavelength offset [nm]} + \text{Fiber laser setpoint [pm]} / 1000$
Piezo modulation (32h)	If the laser module includes the piezo modulation feature, this can be enabled and disabled with the piezo modulation register (32h). Writing the value 0 to the register disables piezo modulation and the value 1 enables it. The format is an 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 as an 8-bit integer.
Temperature compensation mode (35h)	The BasiK modules have a temperature compensation feature 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 it as an 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 receives responses, but does not respond to "write" telegrams. The format is an 8-bit integer.
Fiber laser temperature (11h)	Fiber laser temperature readout in $^{\circ}\text{C}$ ($1/1000\text{ }^{\circ}\text{C}$) in a 16-bit unsigned integer.
Pump current (15h)	Pump current readout in mA in a 16-bit unsigned integer.
Output power (18h)	Output power readout in $1/100\text{ mW}$ in a 16-bit unsigned integer.

Module temperature (19h)	Module Temperature readout in 1/10 °C in a 16-bit signed integer.
Module input voltage (1Bh)	Module supply voltage readout in mV in a 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 (actual) wavelength, add the Wavelength offset as follows:</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 in nanometers, 1-3 nm below the actual wavelength. The wavelength is a constant value for a specific system, as wavelength adjustment is made with the fiber laser setpoint register.</p> <p>The Wavelength offset value is used in conjunction with the fiber laser setpoint value or the Wavelength readout value.</p>

6.6 Koheras BASIK MIKRO Module (K0x2)

6.6.1 General settings

This section describes specific registers for Koheras BASIK MIKRO modules that have product numbers starting with K0x2.

Module type and module address	The module type number is 36h. The standard module address is 1 (01h), but as there can be several modules on one bus, the actual address may be different.
Emission (30h)	The emission register is an unsigned 8-bit integer register. Writing 0 to this register turns off laser emission from the Koheras BASIK Module. Writing 1 to this register turns on laser emission if the Interlock circuit is closed.
Setup bits (31h)	<p>Module setup bits. Refer to the product manual for a thorough explanation of these – 16-bit unsigned integer</p> <p>Bit 0-7: Not used Bit 8: Pump operation constant current Bit 9: Not used Bit 10: Auto start enable (requires FW 1.01 or newer) Bit 11: Not used Bit 12: Enable remote on/off (requires FW 1.09 or newer) – with this bit set to '1' emission can be controlled using 'Module enable' (pin 6 on the IDC connector) Bit 13-15: Not used</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 in a 16-bit unsigned integer.
Output power setpoint, dBm (A0h)	Output power setpoint in 1/100 dBm in a 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 in 1/10 pm in a 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 relate to 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: Not used Bit 3: Not used Bit 4: Module disabled Bit 5: Supply voltage low Bit 6: Module temperature out of range Bit 7: Not used Bit 8: Not used Bit 9: Not used Bit 10: Not used Bit 11: Waiting for temperature to drop Bit 12: Not used Bit 13: Fiber laser temperature settling Bit 14: Not used Bit 15: Error code present</p>
Output power (17h)	Output power readout in 1/100 mW in a 16-bit unsigned integer.
Output power, dBm (90h)	Output power readout in 1/100 dBm in a 16-bit signed integer.
Standard wavelength (32h)	Module wavelength when wavelength offset is 0. Resolution is in 1/10 pm in a 32-bit unsigned integer.
Wavelength offset (72h)	<p>Measured/calculated wavelength offset readout in 1/10 pm in a 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 in a 16-bit signed integer.
Module supply voltage (1Eh)	Module supply voltage readout in mV in a 16-bit unsigned integer.

6.7 Koheras BASiK Module (K1x2)

6.7.1 General settings

This section describes specific registers for Koheras Basik modules that have 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 the register turns off laser emission from the module and writing 1 to it, turns on laser emission if the Interlock circuit is closed.
Setup bits (31h)	<p>These are the module setup bits. Refer to product manual for a thorough explanation of these. The format is a 16-bit unsigned integer.</p> <p>Bit 0: Not used 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: Not used Bit 7: Not used 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 in a 16-bit unsigned integer.
Output power setpoint, dBm (A0h)	Output power setpoint in 1/100 dBm in a 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 in 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 relate to the light source.

6.7.2 Readouts

Status bits (66h)	<p>This register returns the status bits in a 16-bit integer.</p> <p>Bit 0: Emission on Bit 1: Interlock off Bit 2: Not used Bit 3: Not used Bit 4: Module disabled Bit 5: Supply voltage low Bit 6: Module temperature out of range Bit 7: Not used Bit 8: Not used Bit 9: Not used Bit 10: Not used Bit 11: Waiting for temperature to drop Bit 12: Not used Bit 13: Fiber laser temperature settling Bit 14: Wavelength stabilized (X15 modules only) Bit 15: Error code present</p>
Output power (17h)	Output power readout in 1/100 mW in a 16-bit unsigned integer.
Output power, dBm (90h)	Output power readout in 1/100 dBm in a 16-bit signed integer.
Standard wavelength (32h)	Module wavelength when wavelength offset is 0. Resolution is in 1/10 pm in a 32-bit unsigned integer.
Wavelength offset (72h)	<p>Measured/calculated wavelength offset readout in 1/10 pm in a 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 in a 16-bit signed integer.
Module supply voltage (1Eh)	Module supply voltage readout in mV in a 16-bit unsigned integer.

6.7.3 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 in tenths of a percent (permille, ‰) in a 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 in tenths of a percent (permille, ‰) in a 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 in tenths of a percent (permille, ‰) in a 16-bit unsigned integer.

Modulation setup (B7h)

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

Bit 0: Amplitude modulation frequency selector.

Bit 1: Not used

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: Not used

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

Bits 8-15: Not used

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

6.8 BoostiK OEM Amplifier (N83)

This section describes specific registers for BoostiK OEM amplifiers.

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 in a 16-bit signed integer.
Supply voltage (12h)	Supply voltage readout in mV in a 16-bit unsigned integer.
Output power readout (15h)	Optical output power readout in mW in a 16-bit unsigned integer.
Amplifier temperature (17h)	Amplifier temperature readout in 1/10 °C in a 16-bit signed integer.
Pump current readout (19h)	Amplifier pump current readout in mA in a 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the status bits in a 16-bit integer.</p> <ul style="list-style-type: none"> Bit 0: Emission on Bit 1: Interlock off Bit 2: Not used Bit 3: Not used 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: Not used Bit 14: Not used Bit 15: Error code present
Pump current setpoint (21h)	Pump current setpoint when running the amplifier in constant current mode. The setpoint resolution is in mA in a 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 in tenths of a dBm (1/10 dBm) in a 16-bit signed integer.</p>
Off / On / Mode (30h)	<p>Register 30h is used both for switching the amplifier on and off, along with setting the operating mode. Set the value according to the list below.</p> <ul style="list-style-type: none"> 0: Amplifier off 1: Amplifier on, constant pump current 2: Amplifier on, constant output power

6.9 KOHERAS BOOSTIK Line Card (K2x2x)

This section describes specific registers for BoostiK Line Cards.

Module type and module address	The BOOSTIK Line Card module type is 35h. The standard module address is 2 (02h).
Heat sink temperature (1Ch)	Heat sink temperature readout in 1/10 °C in a 16-bit signed integer.
Supply voltage (1Eh)	Supply voltage readout in mV in a 16-bit unsigned integer.
Output power readout (17h)	Optical output power readout in tenths of mW in a 16-bit unsigned integer.
Output power readout monitor (90h)	Optical output power readout in hundreds of dBm in a 16-bit signed integer.
Status bits (66h)	<p>This register returns the status bits as a 16-bit integer.</p> <ul style="list-style-type: none"> Bit 0: Emission on Bit 1: Interlock off Bit 2: Not used Bit 3: Not used 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: Not used Bit 9: Trig in Bit 10: Current limit reached Bit 11: Waiting for temperature to drop Bit 12: Stage 1 current HIGH Bit 13: Stage 2 current HIGH Bit 14: Emission LED bad Bit 15: Error code present
Output power setpoint (22h)	<p>Output power setpoint when running the amplifier in constant power mode.</p> <p>NOTE: The setpoint resolution is in tenths of a mW (1/10 mW) in a 16-bit unsigned integer.</p>
Output power setpoint (A0h)	<p>Output power setpoint when running the amplifier in constant power mode.</p> <p>NOTE: The setpoint resolution is in hundreds of a dBm (1/100 dBm) in a 16-bit signed integer.</p>
Off / On / Mode (30h)	<p>Register 30h is used both for switching the amplifier on and off along with setting the operating mode. Set the value according to the list below.</p> <ul style="list-style-type: none"> 0: Emission off 1: Emission on, constant output power
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 relate to the light source.

6.10 SuperK EXTREME System (S4x2) and SuperK Fianium

6.10.1 Main module

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

Module type (60h) and module address

The module type number is 60h. The standard module address is 15 (0Fh) – 8-bit unsigned integer.

System type (6Bh)

The system type is a newer implementation used to differentiate systems with minor differences. Older versions might not respond to this register. In that case the system type should be interpreted as 0 = SuperK EXTREME. The format is an 8-bit unsigned integer.

0 = SuperK EXTREME

1 = SuperK FIANIUM

Inlet temperature (11h)

The inlet temperature readout is in 1/10 °C in a 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) in an 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 goes through a few intermediate states. This results in brief periods where the returned value differs 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. The format is a 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 connected and in the closed state, the key switch on the front plate is in the On position, and the External bus is terminated with for example 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, you can unset (disable) the interlock (switching interlock relays off) 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) indicates if the interlock circuit is open or closed. The second byte (MSB) indicates 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 division 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 is returned if the ratio is lower than 256, otherwise a 16-bit unsigned integer is returned. This is for historical reasons.

Watchdog interval (36h)

The system can be set to automatically shut-off (laser emission only – not electrical power) in case communication is lost with the host. The value in the watchdog register determines how many seconds without communication the system tolerates before disabling emission. If the register value is set to 0, the feature is disabled. The format is an 8-bit unsigned integer with a maximum value of 255 seconds.

Power level (37h)

Power level setpoint in tenths of a percent (permille, ‰) in a 16-bit unsigned integer.

Current level (38h)

Current level setpoint in tenths of a percent (permille, ‰) in a 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)

This register returns the mainboard status bits as a 16-bit integer.

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

User text (6Ch)

An optional user text register for a SuperK EXTREME system. 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 of the laser, you can select to have the user text written on the top line of the display.

6.10.2 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. The format is an 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. The format is an 8-bit integer.

6.11 RF Driver (A901) for SuperK SELECT

This section describes specific registers for an Internal (within a SuperK EXTREME system) or External RF Driver (A901) together with a SuperK SELECT (A203) accessory. All wavelength and amplitude settings are sent to the RF Driver, not to the SuperK SELECT.

Module type and module address	<p>The RF Driver module type is 66h. For an Internal RF Driver mounted inside the SuperK Extreme chassis, the standard module address is 6. For an External RF Driver, the address depends on the position of the rear panel rotary switch (marked 0..9 and A..F). The actual 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 position of its rotary switch (marked 0..9 and A..F) located close to the external electrical connectors. The actual module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.</p>
RF power (30h)	<p>RF power output of the RF Driver to the SuperK SELECT is controlled with the RF Power register. Writing 0 to this register turns off RF power, whereas the value 1 turns on RF power if the RF Driver has its RF output connected to a RF input on a SuperK SELECT. The format is an 8-bit unsigned integer.</p>
Setup bits (31h)	<p>Register 31h contains three setup bits which control the operation of the RF Driver.</p> <p>Bit 0: Used for 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)	<p>The lowest usable wavelength with the connected crystal. Resolution is in pm in a 32-bit unsigned integer.</p>
Maximum wavelength (35h)	<p>The highest usable wavelength with the connected crystal. Resolution is in pm in a 32-bit unsigned integer.</p>
Crystal temperature (38h)	<p>The measured temperature of the connected crystal in 1/10 °C in a 16-bit signed integer.</p>
FSK mode (3Bh)	<p>A special option. Permits quick switching between different wavelengths. Default value is 0 (FSK off) and standard FSK mode is 3. The format is an 8-bit unsigned integer.</p> <p>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 passes the controls to the Modulation inputs on the RF Driver. Setting the register to 1 disables the Modulation inputs and lets the RF Driver's internal electronics control the RF channels.</p> <p>When using the external Control box (for example, using FSK mode), this register must be set to 0. Default value is 1. The format is an 8-bit unsigned integer.</p>
Connected crystal (75h)	<p>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</p>

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 changes 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, so it can control 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 a percent (permille, ‰). Register 0xB0 controls the amplitude for the first channel, register 0xB1 controls the amplitude for the second channel, and so on. The register formats are 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 in the registers C0h-C7h. Contact NKT Photonics for further information.

6.12 SuperK SELECT (A203)

This section describes specific registers for SuperK SELECT accessories.

Module type and module address	The SuperK SELECT module type is 67h. The address depends on the position of the rotary switch (marked 0..9 and A..F) located close to the electrical connectors. The actual 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 a percent (permille, ‰) in a 16-bit unsigned integer.
Monitor 2 readout (11h)	Readout from optional optical power monitor no. 2 in tenths of a percent (permille, ‰) in a 16-bit unsigned integer.
Monitor 1 gain (32h)	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. The format is an 8-bit unsigned integer. NOTE: Monitor gain settings should not be altered when the SuperK light source is running in external feedback mode (Power Lock).
Monitor 2 gain (33h)	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. The format is an 8-bit unsigned integer. NOTE: Monitor gain settings should not be altered when the SuperK light source is running in external feedback mode (Power Lock).
RF switch (34h)	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. NOTE: RF power must be off before the RF switch position is changed.
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. The format is an 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 in 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 in 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 in 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 in pm.

6.13 SuperK VARIA (A301)

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

Module type and module address	The SuperK VARIA module type is 68h. The address depends on the position of the rotary switch (marked 0..9 and A..F) located close to the electrical connectors. The actual module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.
Monitor input (13h)	Current output power in tenths of a percent (permille, ‰). Requires the optional monitor to be attached for this register content to be valid. The format is 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 (ND) filter setpoint register. The value in this register is in tenths of a percent (permille, ‰).
SWP setpoint (33h)	Short wave pass filter setpoint register in 1/10 nm in a 16-bit unsigned integer.
LWP setpoint (34h)	Long wave pass filter setpoint register in 1/10 nm in a 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the status bits as a 16-bit integer.</p> <p>Bit 0: Not used Bit 1: Interlock off Bit 2: Interlock loop in Bit 3: Interlock loop out Bit 4: Not used Bit 5: Supply voltage low Bit 6: Not used Bit 7: Not used Bit 8: Shutter sensor 1 Bit 9: Shutter sensor 2 Bit 10: Not used Bit 11: Not used Bit 12: Filter 1 moving Bit 13: Filter 2 moving Bit 14: Filter 3 moving Bit 15: Error code present</p>

6.14 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 position of the rotary switch (marked 0..9 and A..F) located close to the electrical connectors. The actual module address is determined by adding 16 (10h) to the hexadecimal setting of the switch.
Wavelength setpoint (31h)	Wavelength setpoint in 1/10 nm in a 16-bit unsigned integer.
Maximum wavelength (32h)	Maximum selectable wavelength in 1/10 nm in a 16-bit unsigned integer.
Minimum wavelength (33h)	Minimum selectable wavelength in 1/10 nm in a 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the status bits in a 16-bit integer.</p> <p>Bit 0: Not used Bit 1: Interlock off Bit 2: Interlock loop in Bit 3: Interlock loop out Bit 4: Not used Bit 5: Supply voltage low Bit 6: Not used Bit 7: Not used Bit 8: Shutter sensor 1 Bit 9: Shutter sensor 2 Bit 10: Shutter sensor 3 Bit 11: Not used Bit 12: Stepper 1 running Bit 13: Stepper 2 running Bit 14: Not used Bit 15: System error code present</p>

6.15 SuperK COMPACT (S024)

This section describes specific registers for SuperK COMPACT lasers that have 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 in a 16-bit unsigned integer.
Heat sink temperature (1Bh)	Heat sink temperature readout in tenths of °C (1/10 °C) in a 16-bit signed integer.
Trig level (24h)	Trig level setpoint in mV. The format is a 16-bit unsigned integer and value range 0 .. 4095 mV.
Display backlight (26h)	Display backlight level in %. The format is a 16-bit unsigned integer and 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). The format is an 8-bit unsigned integer.
Trig mode (31h)	This register is used for setting the SuperK COMPACT operating mode, which is the pulse trig source as an 8-bit unsigned integer. You can set the register as follows:

- 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 connected and in the closed state, the key switch on the front plate is in the On position and the External bus is terminated with for example, 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, you can unset (disable) the interlock (switching interlock relays off) 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) indicates if the interlock circuit is open or closed. The second byte (MSB) indicates 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 as a 32-bit unsigned integer. Value range: Minimum is always 1 Hz. Maximum is system dependent and can be read from Maximum Internal Frequency register.
-----------------------------------	--

Burst Pulses (34h)	<p>This register sets the number of pulses per burst, when using trig mode 2 or 3 using a 16-bit unsigned integer. Value range 1 .. 65535.</p> <p>When emission is on, in trig mode 2, writing to this register initiates a burst. If the telegram contains a value, the 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.</p>
Watchdog interval (35h)	<p>The system can be set to automatically shut-off (laser emission only – not electrical power) in case communication is lost with the host. The value in the watchdog register determines how many seconds without communication the system tolerates before disabling emission. If the register value is set to 0, the feature is disabled. The format is an 8-bit unsigned integer with a maximum value of 255 seconds.</p>
Internal pulse frequency limit (36h)	<p>Readout of the maximum permissible internal frequency in Hz in a 32-bit unsigned integer.</p>
Power level (3Eh)	<p>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. Format is an 8-bit unsigned integer and the value range is 0..100 %.</p> <p>Available in firmware version 1.04, and future versions.</p>
Status bits (66h)	<p>This register returns the status bits as a 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: Not used 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 set bit 10 continuously. About 100 ms after the last trig edge, bit 10 is cleared.</p>
Optical pulse frequency (71h)	<p>Readout of pulse frequency (repetition rate) measured by the system in a 32-bit unsigned integer. Measurement resolution is 2 Hz.</p>
Actual internal trig frequency (75h)	<p>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 as a 32-bit unsigned integer.</p>
Display text (78h)	<p>Readout of the text currently shown in the display. 80 ASCII bytes. Some characters (ASCII values 0 .. 7) are special characters. These may be substituted with whitespace.</p>
Power level (7Ah)	<p>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 in a 8-bit unsigned integer.</p>

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 relate to the light source.

6.16aeroPULSE (P000)

6.16.1 Control Unit

This section describes specific registers for aeroPULSE control units that have product numbers starting with P000.

Module type and module address

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

Inlet temperature (11h)

The inlet temperature readout in 1/10 °C in a 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 the interlock circuit has been reset). If the aeroPULSE System features an additional amplifier based on an aeroGAIN-ROD, the value 5 turns everything on. The register format is an 8-bit unsigned integer.

Reading the register returns the current emission state of the aeroPULSE. Normally, it returns the same value as that which was written. However, while switching emission on or off, the aeroPULSE goes through a few intermediate states. This results in brief periods where the returned value differs from the one that was written.

Setup (31h)

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

- 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 connected and in the closed state, the key switch on the front plate is in the On position, and the External bus is terminated with for example, 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, you can unset (disable) the interlock (switching interlock relays off) 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) indicates if the interlock circuit is open or closed. The second byte (MSB) indicates 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 automatically shut-off (laser emission only – not electrical power) in case communication is lost with the host. The value in the watchdog register determines how many seconds without communication the system tolerates before disabling emission. If the register value is set to 0, the feature is disabled. The format is an 8-bit unsigned integer with a maximum value of 255 seconds.
Power level (37h)	Power level setpoint in tenths of percent (permille, ‰) in a 16-bit unsigned integer.
Current level (38h)	Current level setpoint in tenths of percent (permille, ‰) in a 16-bit unsigned integer.
Status bits (66h)	<p>This register returns the mainboard status bits as a 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.

6.17 SuperK EVO (S1xx, S2xx, S3xx)

6.17.1 Main module

This section describes specific registers for SuperK EVO lasers that have product numbers starting with S1xx, S2xx, and S3xx.

Module type and Module address

There are two different module types. The older module type number is 7Dh, and the current is 8Fh. The standard module address is 15 (0Fh).

NOTE: Some registers differ between the two module types. For module type 8Fh, Ethernet related settings must be directed to a separate Ethernet module, described further below.

Ethernet protocol

TCP

Base temperature (17h)

The Base temperature readout in tenths of °C (1/10 °C) in a 16-bit signed integer.

24V supply (1Dh)

Readout of 24V supply in mV in a 16-bit unsigned integer.

External control input (94h)

Voltage monitoring of the External control input in mV in a 16-bit unsigned integer.

Output power setpoint (27h)

Operating setpoint in ‰ in a 16-bit unsigned integer and with a value range from 0..1000 ‰.

Emission (30h)

Emission on/off. The value 0 turns emission off, and 2 turns it on (if interlock circuit has been reset). The register format is an 8-bit unsigned integer.

Setup bits (31h)

This register is used for setting the operating mode in an 8-bit unsigned integer.

0: Normal Operation
1: External enable activated
2: External feedback activated

Interlock (32h)

If the door interlock is connected and in the closed state, the key switch on the front plate is in the On position, and the External bus is terminated with for example, 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, you can unset (disable) the interlock (switching interlock relays off) 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) indicates if the interlock circuit is open or closed. The second byte (MSB) indicates where the interlock circuit is open, if relevant.

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)	<p>The system can be set to automatically shut-off (laser emission only – not electrical power) in case communication is lost with the host. The value in the watchdog register determines how many seconds without communication the system tolerates before disabling emission. If the register value is set to 0, the feature is disabled. The format is an 8-bit unsigned integer with a maximum value of 255 seconds.</p>
NIM delay (3Bh)	<p>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 and the average step size is 9 ps.</p>
User setup bits (3Dh)	<p>This register is used for user setup in a 16-bit integer.</p> <p>NOTE: Available in module type 8Fh only.</p> <p>Bit 0: Positive trigger output Bits 1..15: Reserved for future use</p> <p>If bit 0 is set to 0, the laser's pulse output is a NIM-style output (negative current). If bit 0 is set to 1, the laser's pulse output is a positive pulse output.</p>
Feedback regulating interval (3Eh)	<p>This register is used for adjusting the feedback regulation speed in an 8-bit unsigned integer.</p> <p>NOTE: Available in module type 8Fh only.</p> <p>NOTE: When using a slow power detector, such as a thermal power meter, it may be necessary to increase this value, to achieve a stable regulation.</p>
Status bits (66h)	<p>This register returns the status bits in a 32-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)	<p>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 relate to the light source.</p>
IP address (B0h)	<p>Sets the system's IP address (static) in an array of four 8-bit values.</p> <p>NOTE: Available in module type 7Dh only.</p> <p>The byte order is the same as the normally written byte order, For example, if the IP address is 192.168.1.17, the data byte order is [C0][A8][01][11].</p>
Gateway (B1h)	<p>The IP address for the gateway in an array of four 8-bit values. The byte order is equivalent to the IP address.</p> <p>NOTE: Available in module type 7Dh only.</p>

Subnet mask (B2h) The subnet mask in an array of four 8-bit values. The byte order is equivalent to the IP address.

NOTE: Available in module type 7Dh only.

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.

NOTE: Available in module type 7Dh only.

6.17.2 Ethernet module

This section describes specific registers for the Ethernet module in the SuperK EVO with module type 8Fh.

NOTE: This module is not available in lasers with module type 7Dh.

Module type and Module address The module type number is 81h. The standard module address is 14 (0Eh).

IP address (B0h) Sets the system's IP address (static) in an array of four 8-bit values.

The byte order is the same as the normally written byte order. For example, if the IP address is 192.168.1.17, the data byte order is [C0][A8][01][11].

Gateway (B1h) The IP address for the gateway in an array of four 8-bit values. The byte order is equivalent to the IP address.

Subnet mask (B2h) The subnet mask in 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) in an array of six 8-bit values. The byte order is equivalent to the IP address.

Port number (B4h) The TCP port number. The default port number is 10001 and the register format is a 16-bit unsigned integer.

DHCP (B5h) Sets the module to operate in either static IP address mode or DHCP mode. The format is an 8-bit integer with the following values:

- 1: Static IP address
- 2: DHCP

Host name (B6h) Sets a host name for the device which is visible on the network. The format is an ASCII string with maximum of 20 characters.

6.18 SuperK FIANIUM (S4x3)

6.18.1 Main module

This section describes specific registers for SuperK FIANIUM lasers that have product numbers starting with S4x3.

Module type (60h) and module address

The module type number is 0088h in a 16-bit unsigned integer.
The standard module address is 15 (0Fh).

Ethernet protocol

TCP

Emission (30h)

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

Reading the register returns the current emission state of the SuperK FIANIUM. Normally, it returns the same value as the one that was written. However, while switching emission on or off, the SuperK FIANIUM goes through a few intermediate states. This results in brief periods where the returned value differs from the one that was written.

Setup (31h)

With the Setup register, the operation mode of the SuperK FIANIUM system can be controlled. The format is a 16-bit unsigned integer with the following values:

0: Internal power control mode
4: External feedback mode (Power Lock)

Interlock (32h)

If the door interlock is connected and in the closed state, the key switch on the front plate is in the On position, and the External bus is terminated with for example, 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, you can unset (disable) the interlock (switching interlock relays off) 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) indicates if the interlock circuit is open or closed. The second byte (MSB) indicates 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 FIANIUM systems featuring the pulse picker option, the division ratio for the pulse picker can be controlled with the pulse picker ratio register. 16-bit unsigned integer.

Watchdog interval (36h)

The system can be set to automatically shut-off (laser emission only – not electrical power) in case communication is lost with the host. The value in the watchdog

	<p>register determines how many seconds without communication the system tolerates before disabling emission. If the register value is set to 0, the feature is disabled. The format is an 8-bit unsigned integer with a maximum value of 255 seconds.</p>
Output level (37h)	Optical output level setpoint in tenths of percent (permille, ‰) in a 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 is an unsigned 16-bit value from 0 to 1023 and the range is 0 – 9.2 ns with an average step size of 9 ps.
User setup bits (3Bh)	<p>This register is used for user setup in a 16-bit integer.</p> <p>Bit 0: Positive trigger output Bits 1..15: Reserved for future use</p> <p>If bit 0 is set to 0, the laser's pulse output is a NIM-style output (negative current). If bit 0 is set to 1, the laser's pulse output is a positive pulse output.</p>
Status bits (66h)	<p>This register returns the mainboard status bits in a 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: Internal temperature out of range Bit 7: Clock battery low voltage Bit 8: Date/time not set ... Bit 13: CRC error on startup (possible module address conflict) Bit 14: Log error code present Bit 15: System error code present</p>

6.18.2 Ethernet module

This section describes specific registers for the Ethernet module in the SuperK FIANIUM (S4x3).

Module type and Module address	The module type number is 81h and the standard module address is 14 (0Eh).
IP address (B0h)	<p>Sets the system's IP address (static) in an array of four 8-bit values.</p> <p>The byte order is the same as the normally written byte order. For example, if the IP address is 192.168.1.17, the data byte order is [C0][A8][01][11].</p>
Gateway (B1h)	The IP address for the gateway in an array of four 8-bit values. The byte order is equivalent to the IP address.
Subnet mask (B2h)	The subnet mask in 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) in an array of six 8-bit values. The byte order is equivalent to the IP address.

Port number (B4h)	The TCP port number. The default port number is 10001 in a 16-bit unsigned integer.
DHCP (B5h)	Sets the module to operate in either static IP address mode or DHCP mode. The format is an 8-bit integer with the following values: 1: Static IP address 2: DHCP
Host name (B6h)	Sets a host name for the device which is visible on the network. The format is an ASCII string with maximum of 20 characters.

6.19aeroPULSE G3 (P4xx, NP4xx)

6.19.1 Main module

This section describes specific registers for aeroPULSE G3 mainboards that have product numbers starting with P4xx, or NP4xx.

Module type and module address

The module type number is 8Bh and the standard module address is 15 (0Fh).

Ethernet protocol

TCP

System type (6Bh)

The system type is used to differentiate systems with minor differences. The format is an 8-bit unsigned integer with the following values:

0 = aeroPULSE G3 FS-50
1 = aeroPULSE G3 FS-50 UV
2 = aeroPULSE G3 FS-20 AOM

Board temperature (11h)

The mainboard temperature readout in 1/10 °C in a 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, the value 5 turns the extra amplifier on. Lastly, if a free space slab amplifier is in the system, the value 6 turns everything on. The format of the register is an 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 goes through a few intermediate states. This results in brief periods where the returned value differs from the one that was written.

Setup (31h)

With the Setup register, the operation mode of the aeroPULSE System can be controlled. The register format is an 8-bit unsigned integer and the possible values are listed below:

4: External feedback mode (Power Lock)

Interlock (32h)

If the door interlock is connected and in the closed state, the key switch on the front plate is in the On position, and the External bus is terminated with for example, 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, you can unset (disable) the interlock (switching interlock relays off) 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) indicates if the interlock circuit is open or closed. The second byte (MSB) indicates 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 aeroPULSE G3 systems featuring the pulse picker option with the SuperK pulse picker, the division ratio for the pulse picker can be controlled with the pulse picker ratio register. The register format is a 16-bit unsigned integer.

**Watchdog interval
(36h)**

The system can be set to automatically shut-off (laser emission only – not electrical power) in case communication is lost with the host. The value in the watchdog register determines how many seconds without communication the system tolerates before disabling emission. If the register value is set to 0, the feature is disabled. The format is an 8-bit unsigned integer with a maximum value of 255 seconds.

Output level (37h)

Output level setpoint in tenths of a percent (permille, ‰) in a 16-bit unsigned integer.

**Femtoplane pulse
picker frequency
(38h)**

For aeroPULSE G3 Systems featuring the pulse picker option with the Femtoplane pulse picker, the output repetition rate of the system can be controlled with the Femtoplane pulse picker frequency register. The repetition rate is in Hz and the register format is a 32-bit unsigned integer.

**Pulse picker NIM
delay (39h)**

For aeroPULSE G3 Systems with the SuperK pulse picker this register sets the NIM trigger delay. For a system with the Femtoplane pulse picker, this register updates the sync output and the NIM delay in one register. The register format is a 16-bit unsigned integer.

**System NIM delay
(3Ah)**

Use this register to set the NIM/Positive logic trigger output on the mainboard in a 16-bit unsigned integer.

User Config (3Bh)

The system can be set up so that the seed trigger output can be set to either NIM standard or positive logic. The register format is a 16-bit unsigned integer and bit 0 sets the output as follows:

0 = NIM trigger output
1 = Positive logic trigger output

**Femtoplane pulse
picker number of
pulses (3Dh)**

For aeroPULSE G3 Systems with the Femtoplane pulse picker, this register sets the number of pulses that are let through the AOMs per repetition. This register only updates when the burst mode is enabled (see register BBh for more details). The register format is a 16-bit unsigned integer.

Stretcher reset (3Fh)

This register is used to reset the stretcher settings to the start of day values. To reset the stretcher a value > 0 must be written to this register. When this register is read it will return 0 when a reset is complete or not in progress. The register format is an 8-bit unsigned integer.

Status bits (66h)

This register returns the mainboard status bits as a 32-bit unsigned integer.

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 8: Date/time not set
 Bit 9: microSD card missing
 Bit 10: Seed calibration necessary
 Bit 11: Seed calibrating
 Bit 12: -
 Bit 13: CRC error on startup (possible module address conflict)
 Bit 14: Log error code present
 Bit 15: System error code present
 Bit 16: Stretcher device active
 Bit 17: Stretcher device ready
 Bit 18: Stretcher setting limits exceeded
 Bit 19: Stretcher setting overrun error
 Bit 20: Stretcher command error
 Bit 21: Stretcher invalid EEPROM
 Bit 22: Stretcher invalid settings profile
 Bit 23: Stretcher set point values unreachable
 Bit 24: AC Okay Failure

Stretcher invalid settings profile (76h)	The stretcher set points can be set in such a way that they can be in a configuration that the stretcher does not support. If this happens, the register reads back a value of 1. Writing a value of 0 to the register in this case clears the error and returns the stretcher set points to the last known good configuration. The register format is an 8-bit unsigned integer.
Stretcher settings unreachable (77h)	The stretcher can have a particular issue where certain configurations of set point values do not flag invalid setting errors, and the stretcher never reaches the desired values and does not settle. If the stretcher has not settled on the desired set point values within 2 minutes, this register reads back a value of 1. Writing a value of 0 to this register clears the error and returns the stretcher set points to the last known good configuration. The register format is an 8-bit unsigned integer.
User text (8Dh)	The aeroPULSE system has a user area register. This is a 240-byte area, which can be used by an operator for identification or other purposes.
Stretcher D2 set point (B1h)	<p>The stretcher set point D2 can be tuned with this register. The D2 setpoint values can be set from -100.0 to +100.0.</p> <p>Writing values into this register does not automatically update the stretcher settings. Once all stretcher set points are set to the desired values use register B5h to send the values to the stretcher. The register format is a 32-bit floating point number.</p>
Stretcher D3 set point (B2h)	<p>The stretcher set point D3 can be tuned with this register. The D3 setpoint values can be set from -100.0 to +100.0.</p> <p>Writing values into this register does not automatically update the stretcher settings. Once all stretcher set points are set to the desired values, use register B5h to send the values to the stretcher. The register format is a 32-bit floating point number.</p>
Stretcher D4 set point (B3h)	<p>The stretcher set point D4 can be tuned with this register. The D4 setpoint values can be set from -100.0 to +100.0.</p> <p>Writing values into this register does not automatically update the stretcher settings. Once all stretcher set points are set to the desired values, use register B5h to send the values to the stretcher. The register format is a 32-bit floating point number.</p>
Stretcher WL offset set point (B4h)	The stretcher set point WL offset can be tuned with this register. The WL offset values can be set from -100.0 to +100.0.

Writing values into this register does not automatically update the stretcher settings. Once all stretcher set points are set to the desired values, use register B5h to send the values to the stretcher. The register format is a 32-bit floating point number.

Stretcher set point update (B5h)

This register is used to send the set point values to the stretcher device. Writing a value >0 starts the set point update in the stretcher. The register format is an 8-bit unsigned integer.

Stretcher current values (B6h)

This register is a compound register that returns all the current values for all four stretcher coefficients. It returns the values in the following order (4 × 32-bit floating point numbers):

Each index is a set of 32-bit floating point numbers.

- 0 = D2 current value
- 1 = D3 current value
- 2 = D4 current value
- 3 = WL Offset current value

Recall stretcher factory set points (B8h)

This register is used to recall one of three factory set point configurations for the stretcher. To use the register, send the value 1, 2, or 3 to recall settings from the factory saved locations in memory. The register format is an 8-bit unsigned integer.

Save user stretcher set points (B9h)

This register is used to save user defined stretcher settings to one of three locations in memory. Once the set points registers are set up as the user would like to save, send either a 1, 2, or 3 value to this register and the set points will be saved to the specified location. The format is an 8-bit unsigned integer.

NOTE: This register saves the values in the set point registers, make sure that those values have been sent to the stretcher using register B5h and that they produce the output required before saving the setting.

Recall user stretcher set points (BAh)

This register is used to recall one of three user set point configurations for the stretcher. To use the register, send the value 1, 2, or 3 to recall settings from the user saved locations in memory. The format is an 8-bit unsigned integer.

Femtoplane burst mode enable (BBh)

For aeroPULSE G3 Systems with the Femtoplane pulse picker, this register is used to enable/disable burst mode on the laser output. With this feature enabled register 3Dh will accept changes to number of pulses per repetition rate. The format is an 8-bit unsigned integer.

Shutter enable (E6h)

This register allows the user to open and close the mechanical shutter manually. If the shutter is closed while the laser is on the emission stays on ready for when the shutter is reopened. Writing a value of >0 opens the shutter and the register reads 1 when it is open and 0 when closed. The format is an 8-bit unsigned integer.

Stretcher auto-calibration (E7h)

This register enables the stretcher auto-calibration. Writing a value of 1 to this register starts the auto-calibration. When disabling the feature, and the stretcher is only partially optimized, the calibration continues until it has finished its current run through before disabling. The format is an 8-bit unsigned integer.

The register reports as follows:

- 0 = auto-calibration off
- 1 = auto-calibration on
- 2 = auto-calibration in progress

NOTE: This feature is designed to hold a lock and slightly tune from a known good set of parameters and should not be used if the settings have been detuned.

6.19.2 Femtoplane Pulse Picker

This section describes specific registers for the Femtoplane pulse picker module that is present in aeroPULSE G3 systems.

Module type and module address	The module type number is 79h and the standard module address is 4 (04h).
AOM2 number of pulses/bursts (34h)	This register is used to set the number of pulses or bursts (if burst mode is enabled) let through the AOM2 per trigger event. The register is linked to trigger mode register 77h. If trigger mode is set to 0 then this register will return 0 and not allow the value to be updated. See trigger mode register 77h for an explanation on what modes are available. The register format is a 16-bit unsigned integer.
Software trigger (39h)	This register is used to reset AOM2 and start a new trigger event. Writing a value >0 starts a software trigger event. The register format is an 8-bit unsigned integer.
Number of pulses limit (75h)	This register returns the maximum number of pulses that are allowed when burst mode is enabled. The value in this register adjusts with the repetition rate, such that as the repetition rate increases the number of pulses allowed decreases. The maximum number of pulses that is allowed at lower repetition rates is 15. The register format is a 16-bit unsigned integer.
Trigger mode (77h)	<p>The trigger mode register is used to keep track of what trigger mode is in use. There are three possible options from 0 to 2. When option 0 is set, the pulse picker is set to free-running mode which makes the pulse train repeat continuously without stopping. Mode 1 and 2 sets the pulse picker to only let through the number of pulses/bursts that are set in register 34h. The register format is an 8-bit unsigned integer.</p> <p>The modes are as follows:</p> <ul style="list-style-type: none"> 0 = Free-running 1 = External trigger 2 = Software trigger
Trigger buffer enable (78h)	This register is used to enable/disable the buffer for the trigger input. If external trigger is to be used this register is required to be enabled. Writing a value >0 to the register enables the buffer. The register format is an 8-bit unsigned integer.
Gate buffer enable (79h)	This register is used to enable/disable the buffer for the gate input. If gate input is to be used this register is required to be enabled. Writing a value >0 to the register enables the buffer. The register format is an 8-bit unsigned integer.
AOM2 output power (8Ch)	The AOM2 output power level setpoint in tenths of a percent (permille, ‰). This register only affects the output power if the external analogue control register B2h is set to 0. The register format is a 16-bit unsigned integer.
External analogue control (B2h)	This register is used to switch between internal analogue and external analogue control for the output power of the system. Writing a value >0 will switch to using external analogue input to control the AOM2 output power. If the register is set to 0 the AOM2 output power is controlled by the AOM2 output power register 8Ch. The register format is an 8-bit unsigned integer.

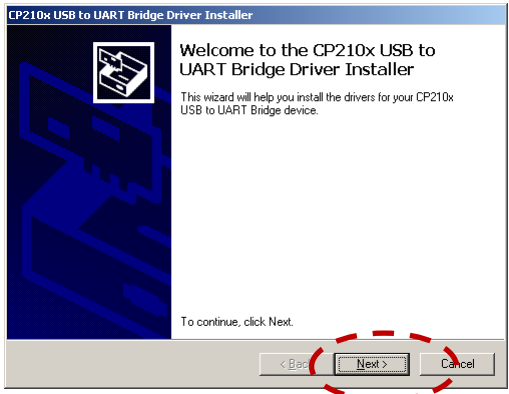
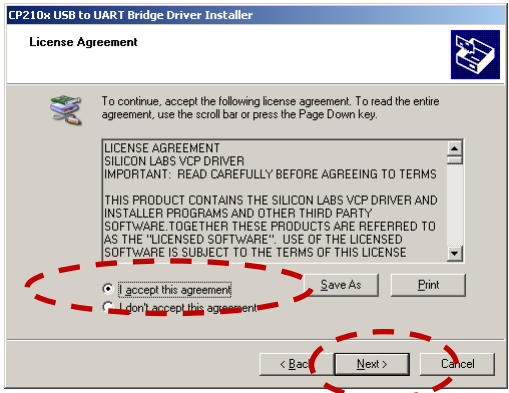
7 Silabs USB Driver

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

USB Driver

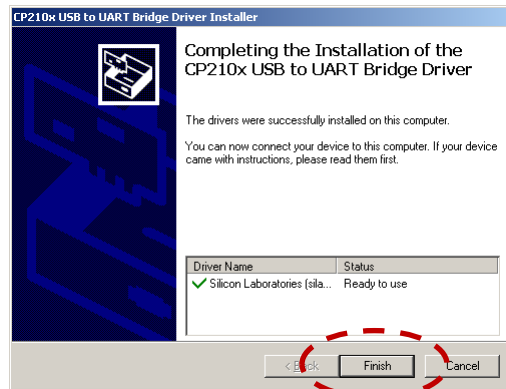
The following procedure describes how to install the driver software.

NOTE: By default, the driver is installed by the SDK installer, 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 to assist programmers to verify and understand how to communicate with NKT Photonics products. The Generic User Interface can control all NKT Photonics products communicating via the NKT Photonics Interbus. For example, using a PC running Windows you can control Koheras Basik modules and SuperK EXTREME lasers.

LabVIEW Runtime Engine

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

8.1 Installing the software

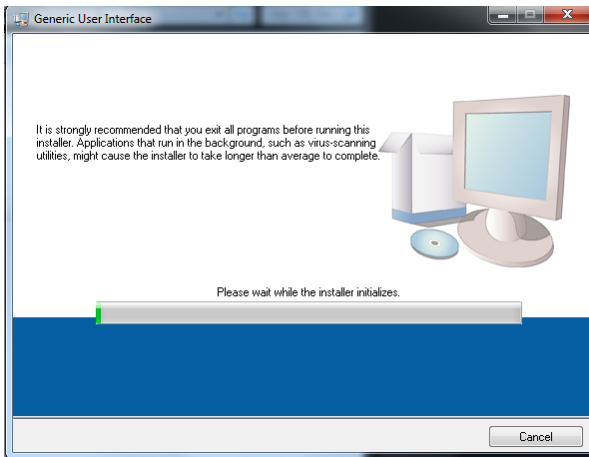
USB

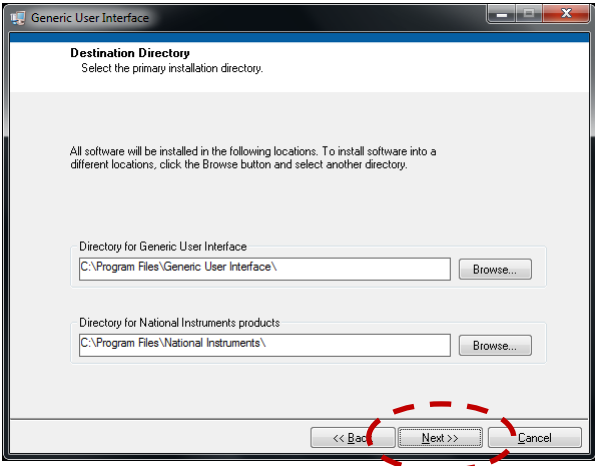
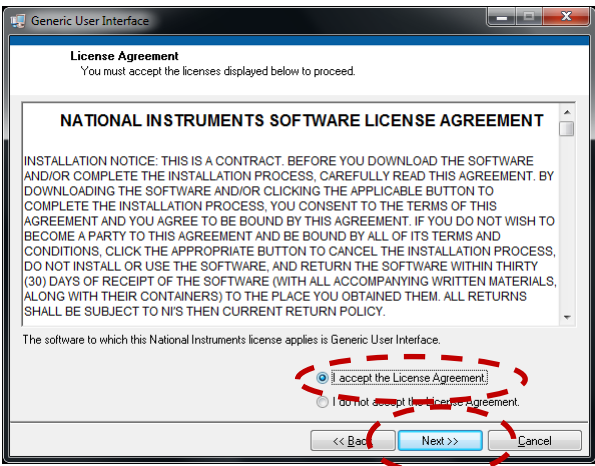
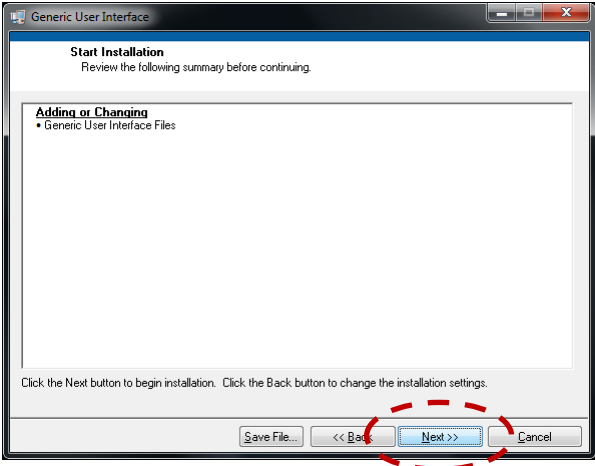
For information about how to install USB driver refer to [Silabs USB Driver](#)

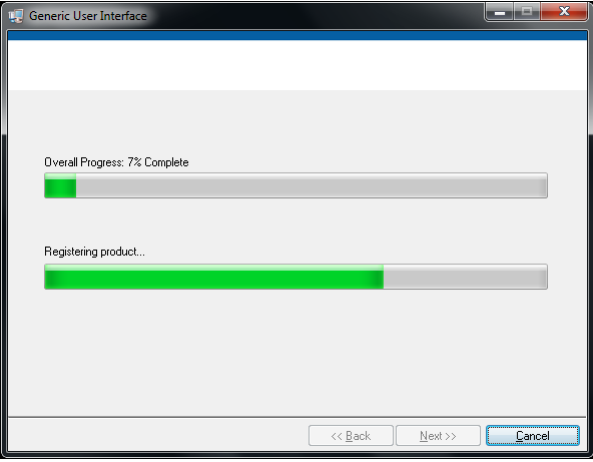
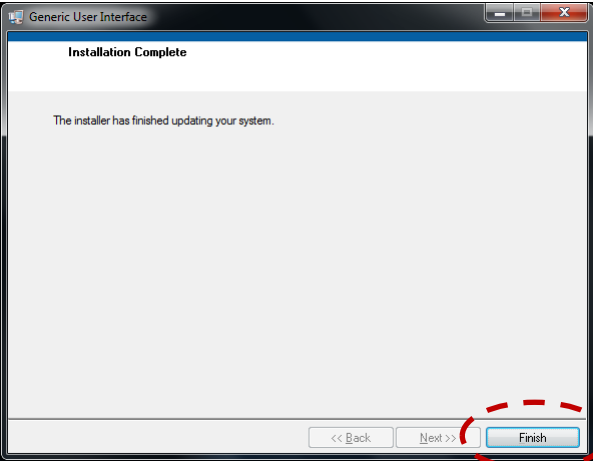
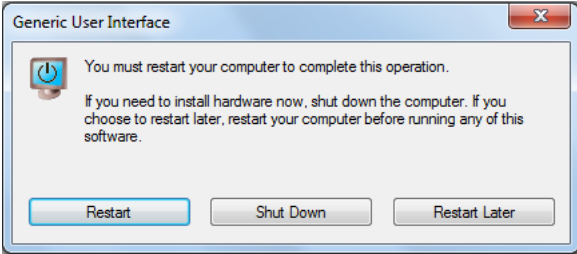

Generic User Interface Software

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

NOTE: The Generic User Interface is installed by the SDK installer by default. However if you opted not to install it during the SDK install, you can install it according to the following procedure.

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 is displayed.</p> 

4	<p>The installer requests that you select the installation directories. Choose the suggested directories by clicking Next.</p> 
5	<p>Read the License Agreement, and choose <i>I accept the License Agreement(s)</i> and click Next to proceed.</p> 
6	<p>Click Next to start copying files to the computer.</p> 

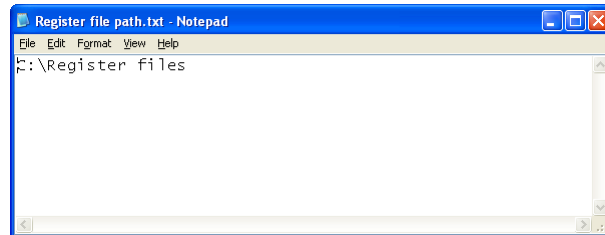
7	<p>A new window shows the progress of the installation.</p> 
8	<p>When the installation is complete, click <i>Next</i> to end the installer.</p> 
9	<p>After installation of the software, the computer may have to be restarted. In this case, click on the <i>Restart</i> button to restart the computer.</p> 
10	<p>In the Start > Programs folder you will find a shortcut to the Generic User Interface. Click on this shortcut to launch the 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 determines the module's type number. The module type number is then used to locate the corresponding register file, and from this the Generic User Interface can show the 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 you can change this in the text file if the files are located in another folder.



8.3 Using the Generic User Interface Software

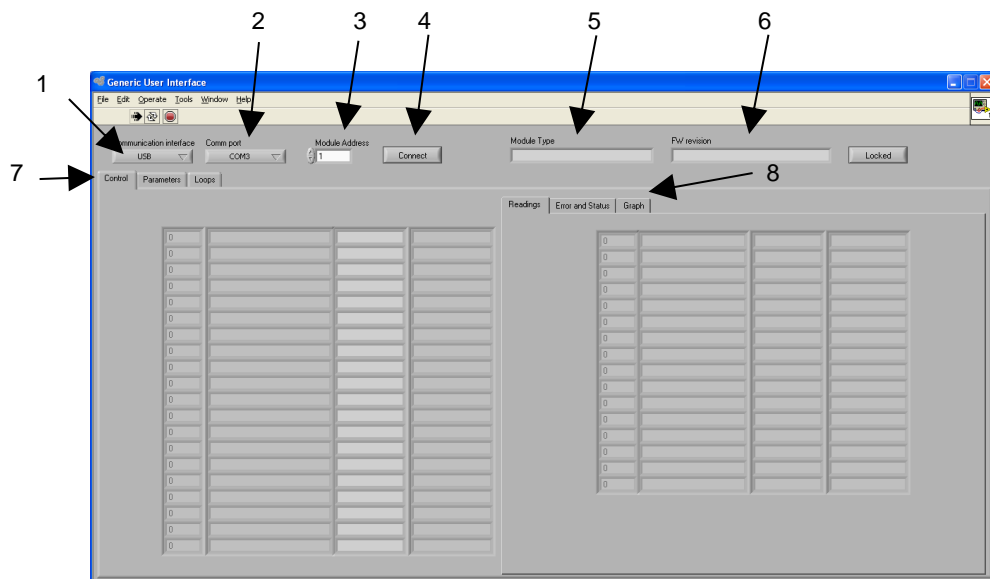
Introduction

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

The front panel interface is described in the following:



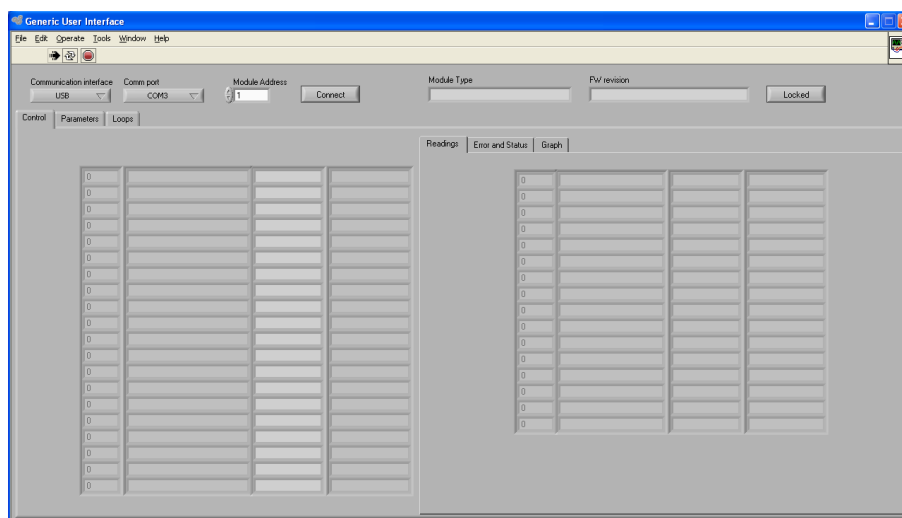
1 – Communication Interface	Select either USB or Ethernet.
2 – Comm Port or IP Address	<p>For USB, choose the COM-port to communicate over. A COM-port in the list marked with *, indicates a known type of communication port, so this is likely the COM-port to choose.</p> <p>For Ethernet, type in the IP address to communicate with.</p>
3 – Module Address	Enter (or use the up/down buttons to select it) 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, <i>Connected</i> is displayed. Click the button again to end the connection and close the communication port.</p>
5 – Module Type	In this text field the Generic User Interface displays 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 displays the current firmware revision of 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"> In 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"> In 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 choose between 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.

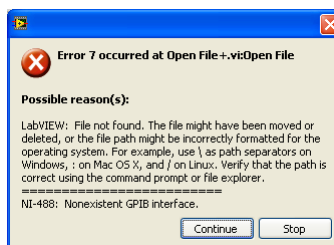


If USB is used, then select the COM port number for the connection. An asterisk next to the COM port in the drop down menu indicates that it is a known Communication interface, so this is likely the COM port to choose. If Ethernet is used, type in the IP address for the system to communicate with.

Choose the Module Address to communicate with. The Module Address can be changed anytime even if the Generic User Interface is already connected.

Register File Path

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



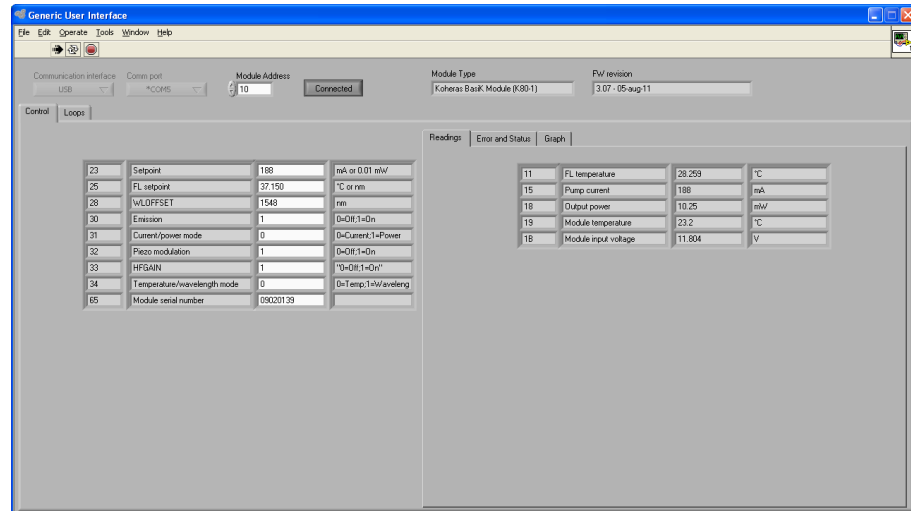
In this case, click Stop and define the correct path in the Register file path.txt file as described in the [Register File Path](#) section.

Click on the Connect button to try to establish communication between computer and a connected 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 you can read out and write new values to the input registers.



The first column of the array shows the register address as a hexadecimal number. The second column shows a small description of the register and the fourth column shows the units used for each register. The content of these three columns is taken directly from the actual register file.

Register Values

The third column shows read outs from each register. The Generic User Interface handles the scaling of each register.

Notice

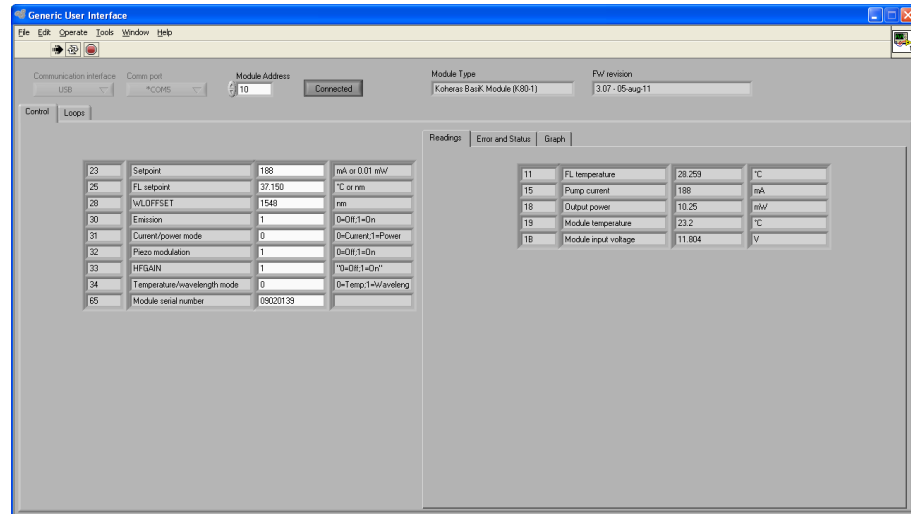
The third column is a string array, even if values are numbers. So the Enter button on the keyboard should not always be pressed when a new value has been typed into the array. Instead click with the mouse somewhere on the grey window background to write the value to the module.

Warning

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

8.3.4 Readings

In the Controls pane, the Readings sub-pane can be selected. In this pane all read-only registers of the chosen module/system are read out.



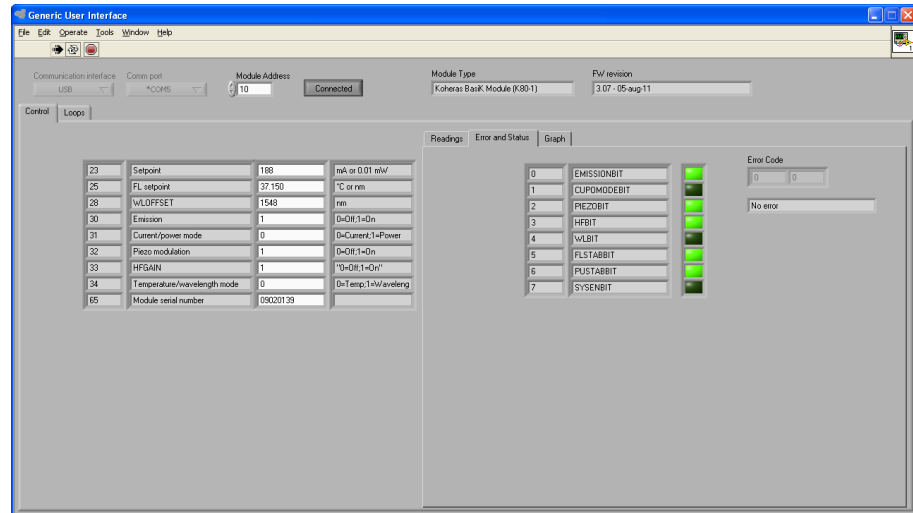
Similar to the Control registers, the Reading array contains four columns. The first column shows the hexadecimal number for the register addresses and the second column shows a short description of the register. The third column shows the readings from the module/system and the fourth column shows the units used for the values.

Columns 1, 2 and 4 are taken directly from the Register File and column 3 has the actual values read from the module/system.

Similar to the Control registers, the Generic User Interface handles the scaling of the registers.

8.3.5 Error and Status

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



Status Bits

The Status Bits are 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.

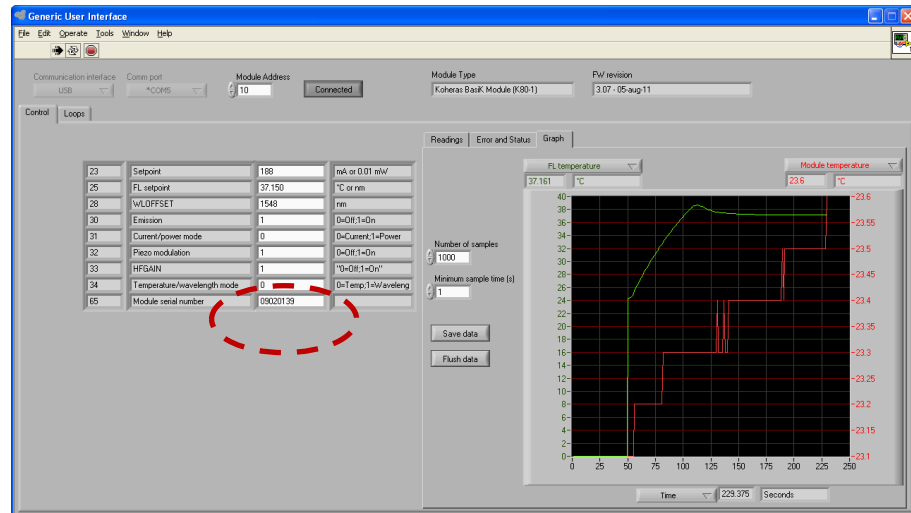
Error Code

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 beneath the Error Code field.

The description of each Status Bit and Error Code is taken from the register file.

8.3.6 Graph

In the Controls pane, the Graph sub-pane can be selected. In this pane the content from both the Readings array and the Control array can be visually represented in a graph.



X-axis

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

If Time is chosen the graph shows a display 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 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. You can choose the same parameters that are available for the X-axis.

The left-hand Y-axis uses green text along the gridline corresponding with the green curve plotted 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. You can choose the same parameters that are available for the X and Y-axis.

The right-hand Y-axis uses red text along the gridline corresponding with the red curve plotted in the graph.

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 for example, I-P curves (dependant on modules/systems).

Number of Samples

In this field, you can select the maximum number of data points that are shown in the graph. Right after the Generic User Interface has connected to a module/system there are 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 are erased from the computers memory.

Minimum Sample Time

The Minimum Sample Time determines the minimum time between register values being updated. It is as it says a minimum value, so the actual time is slightly larger. Setting the Minimum Sample Time to 0, makes the Generic User Interface communicate as fast as possible.

Save Data

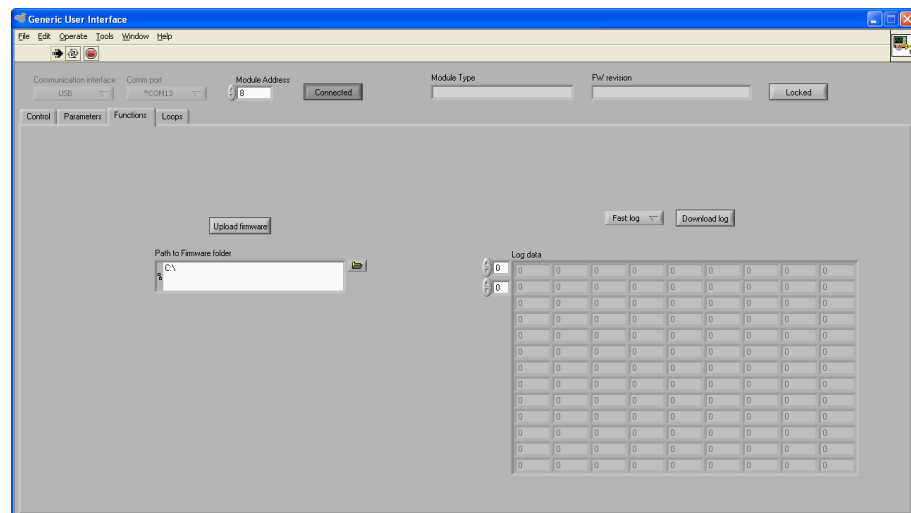
The Save Data button makes it possible to save all recorded data to a tab-delimited ASCII-file. All recorded data means not only what is shown on the graph, but also all other parameters that are not selected to graph.

Flush Data

Clicking on the Flush Data button erases all recorded data, and the graph starts 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. These are available on the Functions pane if they are supported by the module/system that the Generic User Interface is communicating with.



Upload Firmware

To upload new firmware to the module the Generic User Interface is connected to, browse to the directory where the desired hex-file to be uploaded is located. Once the folder path is selected, click on the Upload Firmware button to start the upload. During the upload, the button shows Uploading Firmware.

Download Log

If the module the Generic User Interface is connected to includes the module log-function, the log file(s) can be downloaded with the Generic User Interface.

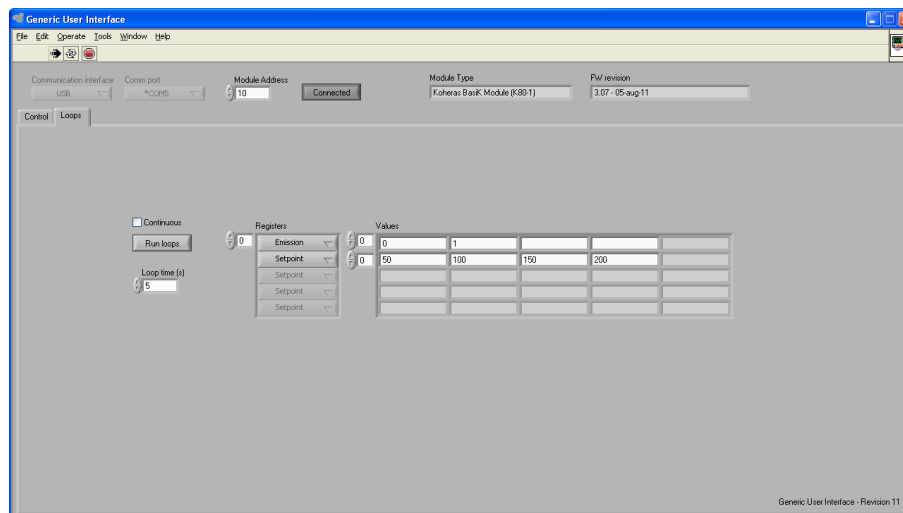
If more than one log-file is available, the log-file to be downloaded is selected with the selector to the left. Click on the Download Log button to begin the actual download. The log-files can collect large amounts of data over time, so expect a long duration to download these log-files.

8.3.8 Loops

Loops

In the Loops pane it is possible to let the Generic User Interface automatically execute a defined sequence.

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



Registers

In the Registers array the parameters for the sequence to step through are 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

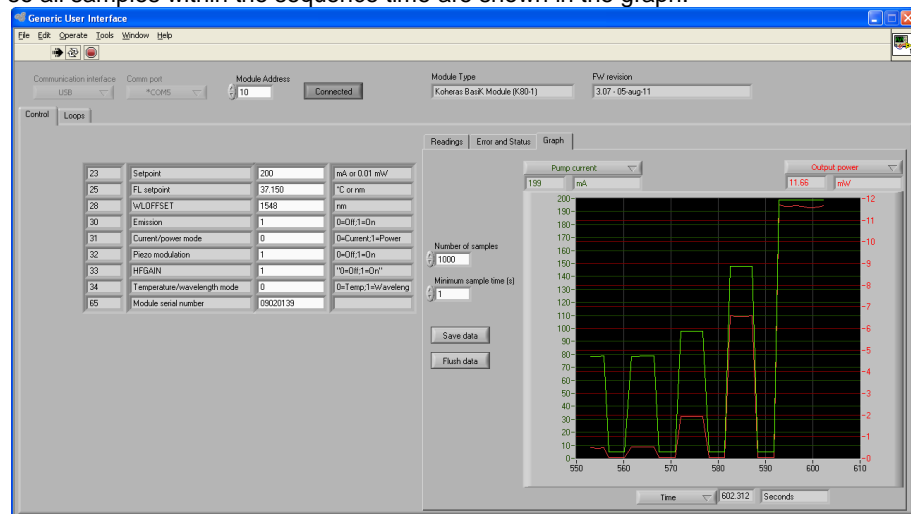
If the Continuous check mark is set, the Generic User Interface will run the sequence over and over again.

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

In 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 are 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 a digital interface, based on the NGSerialPort VI from NKT Photonics.

The functions used in this chapter are included for compatibility reasons. We recommend that any new development be 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

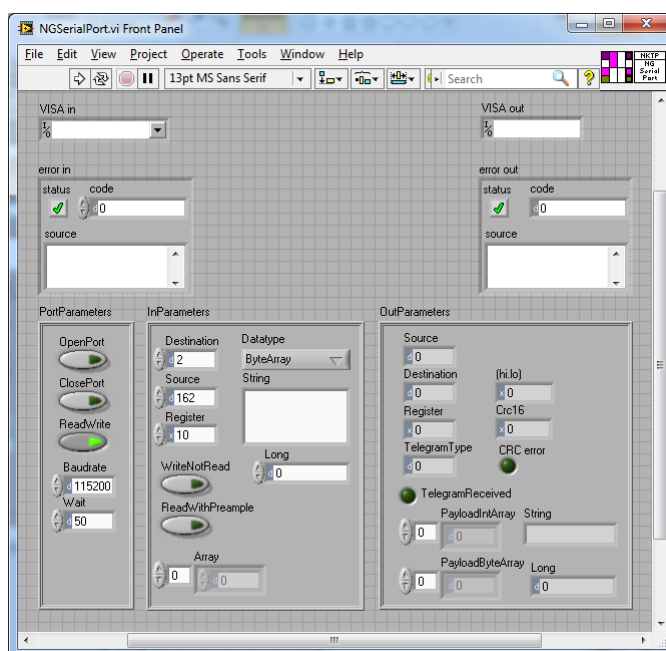
The NGSerialPort VI contains all functions needed for establishing communication from LabVIEW running on a computer connected to an NKT Photonics device.

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.

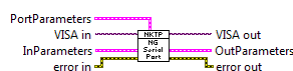
9.1.1 Inputs and Outputs

This section describes the inputs and outputs of 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, and is used to configure which COM-port is used for communication. Notice that even if the module(s)/system(s) are connected to the computer with an USB connection, this connection establishes a virtual COM-port, which should be used for the communication.

VISA out

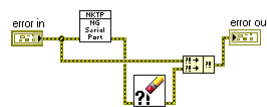
This terminal is an output, which provides the COM-port number used for communication. This can be connected to the following NGSerialPort in the block diagram if desired.

error in

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

error out

A typical error output cluster. It can be connected to the following function with an *error in*. However, notice that if write commands are sent to input registers on a module/system that does not reply with an acknowledge (ACK) or not acknowledge (NACK) the NGSerialPort VI times out and generates 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.

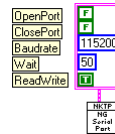


Modules and systems that do not by default generate ACK and NACK are:

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

9.1.2 PortParameters

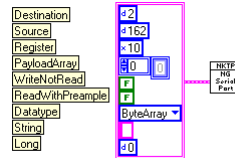
An input cluster to setup, open and close the communication port and to select whether the VI should be used for communication. This input cluster can either be configured with the bundle function in LabVIEW or a constant as shown below.



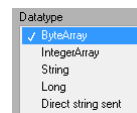
OpenPort	The communication port does not have to be opened every time LabVIEW is communicating with the module/system. The communication port must be opened the first time (or prior) and again after the communication port has been closed if further communication is desired.
ClosePort	When no further 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 is left 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 communicate 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 wait for a response from the interfacing module/system. Typical values are 50 or 100 milliseconds.
ReadWrite	The ReadWrite input is selected if the NGSerialPort VI is used for actual communication and not only for opening or closing the communication port. Whether the VI is used for writing or reading, this input should be set high.

9.1.3 InParameters

An input cluster with information about addresses and whether the VI should be used for reading or writing. In case it is used for writing, information on what type of data is written and the actual data to be written. This input cluster can either be configured with the bundle function in LabVIEW or as 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 set higher than 160 (A0h).
- Register** This input configures the register address to read or write.
- Datatype** In case data is 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 is transmitted.

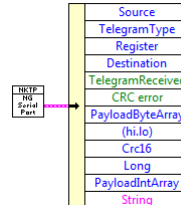


If just an integer is transmitted, the Long function can be used.

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

9.1.4 OutParameters

An output cluster with information about addresses and whether the VI is used for reading or writing. In case it is used for writing, information on what type of data is written and the actual data to be written. The output can be split-up with the Unbundle By Name function as shown below.



Source This output provides the sender address of the received telegram in a one-byte decimal value.

TelegramType The TelegramType output provides general information about the received telegram. (This is the same as the message type described in section 2.2.)

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 into a register.
8	Datagram	Response. Register content returned; caused by a previous "Read".

Register This output provides the register 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. The value type is a one-byte decimal value.

TelegramReceived A 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 verifies the CRC value. In case the telegram was corrupted during the transmission, it reports 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. This is a two-byte hexadecimal value.
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 brief example of how to use the NGSerialPort from the front panel. The example shows how to read out the wavelength #0 setting from an RF driver that feeds an RF signal to a SuperK Select and how 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.

NOTE: In a final LabVIEW application when calling the NGSerialPort multiple times, it is not required to open and close the communication port every time, only ensure 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 an external RF driver is sends an RF signal to the SuperK Select. The Interbus address of the RF driver is determined by a rotary switch on its chassis which is set to position 3. Adding 0x10 to the number (3) which the rotary switch is set to, sets the actual module address to 0x13. This number is used in the Destination address field.

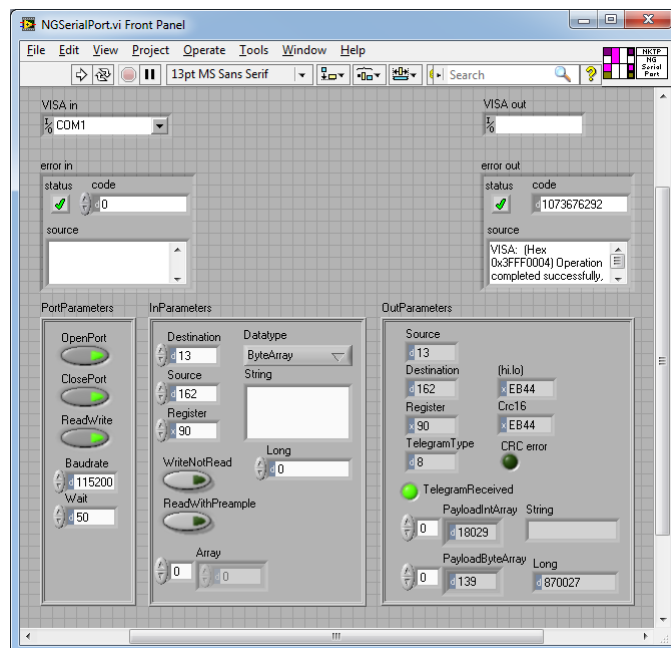
The Source address field is set to address 162.

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

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

Run

Run VI by clicking on the arrow in the top left corner of the Front Panel.



OutParameters

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

The received telegram is presented in both 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

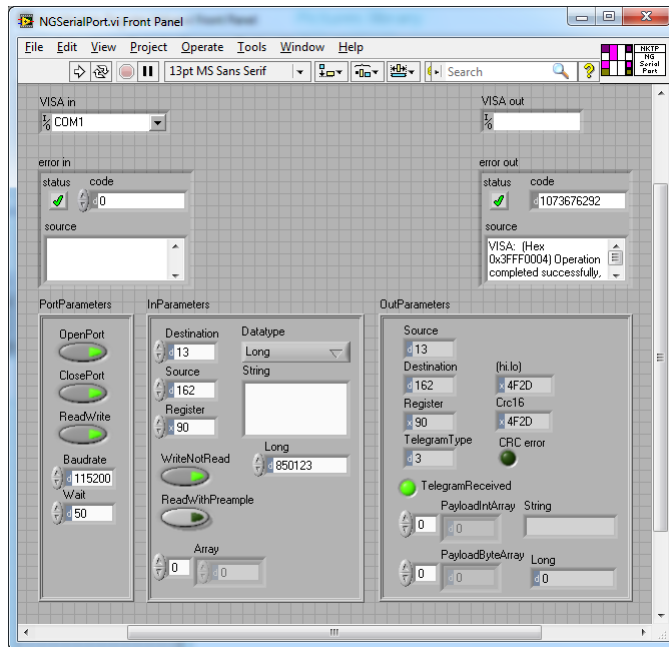
To write a new wavelength to the wavelength #0 register, change WriteNotRead to high.

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

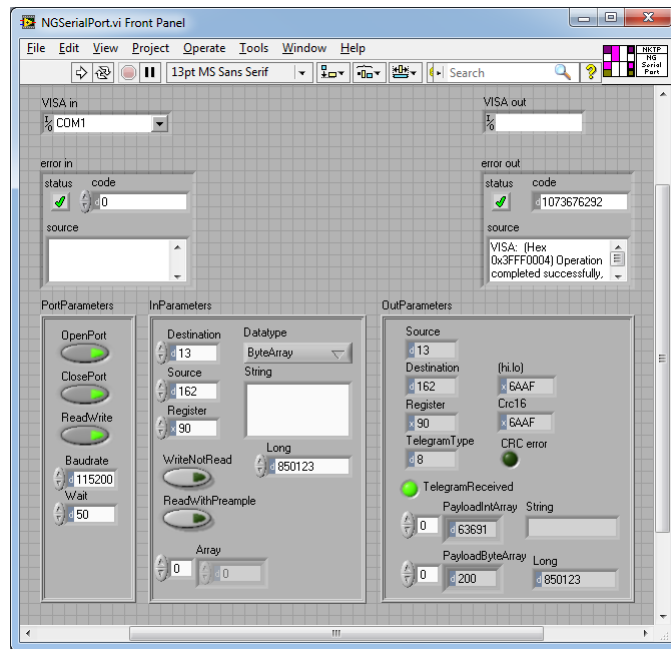


OutParameters

When the RF driver receives this command, it replies 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 leave all other settings unchanged to read out the new setpoint by running VI again.



In the Long field of the OutParameters, the new setpoint is displayed.

9.1.5.3 Writing other types of registers

- Byte(s)** 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)** To transmit a 16-bit integer or integer array, set the Ring selector to IntegerArray. Type in a single integer into the first cell of the Array, or use following cells in the Array for multiple integers.
- String** To write a string 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 previous example. To write a 32-bit long array to a module, split it into a 16-bit array with the least significant 16-bit value first, followed by the most significant value (illustrated in the following figure).

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)	...

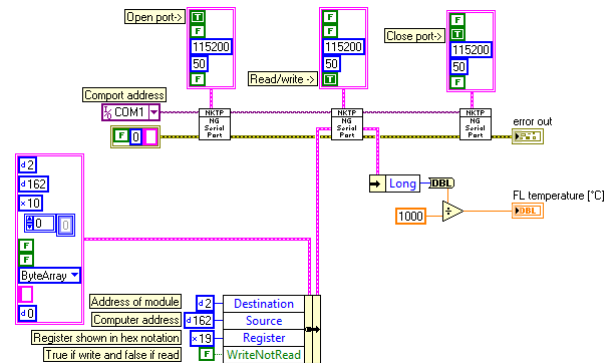
9.1.6 Programming Examples

This section provides two programming examples:

- an example for reading a value from a module.
- an 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 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 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 NGSerialPort VI is called a second time, the PortParameters are set to read/write.

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

Closing Port

Lastly before ending the program, the port is closed by setting the second boolean value high in the PortParameter cluster for the third call from 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 and described as follows.

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