

BasiK integration guide
Version 1.0
Issue date: 15/10-2008

Index:

Protocol description	3
Physical	3
Telegram.....	3
Framing.....	3
Message	3
Header.....	3
Payload	3
CRC16	4
Transmitting and receiving.....	5
Escape sequence conversion.....	5
Transmission example	6
Reception	7
Reception example	7
Example of communication	8
Basik module hardware integration	9
BasiK module register definition.....	10

This document describes the Koheras protocol and Basik module integration.

Protocol description

Physical

The Koheras Common Bus standard is based on a 2 wire RS485. Baudrate 115200 kbit, 8 data bits, 1 stop bit, No parity.

Telegram

Framing

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

[SOT][MESSAGE][EOT]

SOT: 13 0D hex
EOT: 10 0A hex

Message

A message consists of a header, a payload and cyclic-redundancy-check (CRC).

[HEADER][PAYLOAD][CRC16]

Header

The header consists of 2 bytes, a destination and a source address.

[DESTINATION][SOURCE]

Destination: 1 byte, 0..255
Source: 1 byte, 0..255

Payload

The payload may consist of 0 to 255 bytes of data.

The first byte of the payload is a command byte. Number 0 to 8 are commands reserved for the communication and special actions.

Number	Command
0	Nack
1	CRC error
2	Busy
3	Ack
4	Read
5	Write
6	HALT!
7	Relay
8	Datagram

Values higher then 255 (more the one byte) is transmitted with LSB first.

CRC16

The CRC16 value consists of two bytes calculated on the complete message (header and payload).

The CRC16 calculation is preformed before escape sequence conversion on the outgoing telegram.

The CRC16 value is transmitted with MSB first.

Initial CRC16 value is 0.

Below is an example how to implement CRC16 in C. Char is a 8bit value, int is a 16bit value.

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

```
unsigned char ser_data;  
static unsigned int crc;
```

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

Transmitting and receiving

Escape sequence conversion

Before a message is transmitted, it must be escape sequence converted. If a data byte is equal to one of the four numbers 10, 13, 64 or 94 a special action must take place.

A start of escape sequence character is inserted in the message and escape sequence character (ECC) is added to the data byte.

Below is a small C example how to make the escape sequence conversion. The function "com1_putc" handles the transmission by the use of the UART.

```
void TXnocrc (unsigned char data)
{
    if(data == SOT || data == EOT || data == SOE)
    {
        data += ECC;
        com1_putc(SOE);
    }
    com1_putc(data);
}
```

Name	Number (Decimal)	Number (Hexadecimal)	Meaning
EOT	10	0A	End of telegram
SOT	13	0D	Start of telegram
ECC	64	40	Escape sequence character
SOE	94	5E	Start of escape sequence

Ex.:

When the following sequence of data is transmitted

[42][15][10][187][94][11], the two data bytes 10 and 94 must be converted.

After escape sequence conversion:

[42][15][94][10+64][187][94][94+64][11]

[42][15][94][74][187][94][158][11]

And the final telegram with framing will be:

[13][42][15][94][74][187][94][158][11][10]

Transmission example

Below is an example with all functions to transmit a complete telegram

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

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

void TXnocrc(char data)
{
    if(data == SOT || data == EOT || data == SOE)
    {
        data += ECC;
        com1_putc(SOE);
    }
    com1_putc(data);    // Sends data to UART
}

unsigned int tx1(char data, unsigned int crc)
{
    crc = CRC_add1(data,crc);
    if(data == SOT || data == EOT || data == SOE)
    {
        data += ECC;
        com1_putc(SOE);
    }
    com1_putc(data);
    return crc;
}

// Dest is the destination address, size is number of data bytes to be transmitted, cmd is the
// command byte and *data is a pointer to the data. MY_NODE is the address of the transmitter.
// DIR1 is a hardware pin to control direction of RS485 interface.

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

    DIR1 = OUTGOING;    // Change direction of RS485
    com1_putc(SOT);    // Start a new telegram
    crc_value = tx1(dest, 0);    // Transmitting destination address
    crc_value = tx1(MY_NODE, crc_value);    // Transmitting source address
    crc_value = tx1(cmd, crc_value);    // Transmitting command
    for(tt=0;tt<size;tt++)
    {
        crc_value = tx1(*data,crc_value);    // Transmitting data
        data++;    // Increment datapointer
    }
    TXnocrc((crc_value>>8) & 0xFF);    // high byte of CRC
    TXnocrc(crc_value & 0xFF);    // low byte of CRC
    com1_putc(EOT);    // End telegram
    DIR1 = INGOING;    // Change the direction of RS485
}
```

Reception

Reception of a telegram is done in reverse order of a transmission.

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

When a SOE is received subtracts ECC (64) from the next data byte and calculate the new CRC.

When everything else then SOT, EOT, ECC or SOE is received calculate the new CRC.

When an EOT is received CRC must be 0 otherwise a CRC-error has occurred.

Below is a C example how to implement the reception, reverse escape sequence conversion and CRC check. Some additional checks may be implemented to catch over run error etc.

Reception example

```
short TelegramReady, Inframe, Escape_seq1; // boolean
char rcv_counter, RcBuffer[BUFLNGTH], *RcInPtr1;
unsigned int crc1;

void COM1_isc(void)      //Called by interrupt
{
    char ch;
    while(UARTnotEmpty)
    {
        if(!FERR1)                // no Framing Error
        {
            ch = RcREG1;           // Get byte from UART
            switch(ch)
            {
                case SOT:           // New telegram
                    rcv_counter1 = 0;
                    inframe1=1;
                    crc1 = 0;
                    RcInPtr1 = &RcBuffer[0];
                    TelegramReady = 0;
                    break;
                case EOT:           // End of telegram
                    inframe1=0;
                    if(CRC1 == 0) TelegramReady = 1; // CRC ok and telegram is ready
                    break;
                case SOE:           // start of escape sequence
                    Escape_seq1 = 1;
                    break;
                default:            // Data byte
                    if(inframe1==1)
                    {
                        if(Escape_seq1 == 1)
                        {
                            Escape_seq1 = 0;
                            ch -= ECC;
                        }
                        crc1 = CRC_add1(ch, crc1);
                        *RcInPtr1 = ch;
                        if(rcv_counter1 < BUFLNGTH)
                        {
                            RcInPtr1++;
                            rcv_counter1++;
                        }
                    }
                    break;
            }
        }
    }
}
```

Example of communication

This example shows the communication between an external device (PC, embedded system, micro controller etc.) and the Basik module.

Telegram for reading Basik module status and warning register.

This Basik module has the address 09 hex

The PC or the unit communicating with the module has the address 42 hex, any value higher then 31 is legal.

Status and warning register is 1F hex.

Read command is 04 hex.

All values are in hexadecimal.

[0D][09][42][04][1F][AF][A0][0A]

The BasiK module responds with

[0D][42][09][08][1F][63][00][FD][C4][0A]

42 = Destination

09 = Source

08 = Datagram

1F = Status and warning register

63 = Emmission on, Constant power mode, Piezo tuning off, No RIN suppression, Temperature tuning

00 = No warnings

FDC4 = CRC16 for the telegram

Telegram for switching Basik module on.

This Basik module has the address 09 hex

The PC or the unit communicating with the module has the address 42 hex, any value higher then 31 are legal.

Emission control register is 30 hex.

Write command is 05 hex.

All values are in hexadecimal.

[0D][09][42][05][30][01][D5][CC][0A]

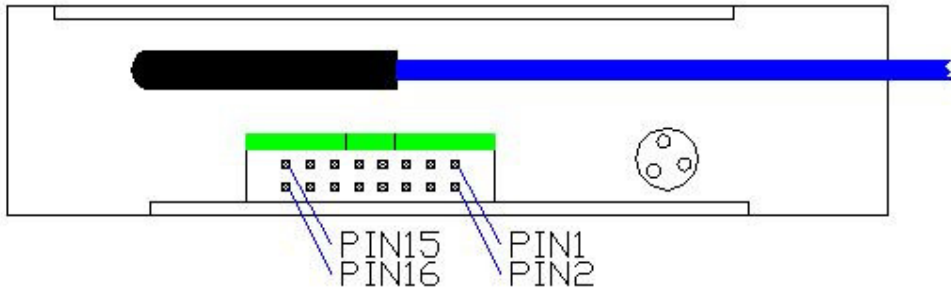
Telegram for switching Basik module off.

[0D][09][42][05][30][00][C5][ED][0A]

Basik module hardware integration

Connector type

To connect to the BasiK module use IDC-16 female (Ex.: Molex 90635-1162)



Pin description

1	Not used
2	Emission indicator logic output, High = Emission, Low = No emission
3	RS485 D-
4	RS485 D+
5	Interlock loop+
6	General system enable, logic input, must be pull high with a 4.7kΩ resistor to 5V or with 10kΩ resistor to 12V.
7	Interlock loop-
8	Interlock, logic input, must be pull high though high with a 4.7kΩ resistor to 5V or with 10kΩ resistor to 12V.
9-12	GND
13-16	+12V

BasiK module register definition

Attention: Do not read or write BasiK module faster then 50 times pr. sec.

Reading: [Destination][Source][04][Register][CRC msb][CRC lsb]

Register (Hex)	Description
10	Reading this register returns all internal measurements including status and warnings, 13 x 2 bytes + 2 bytes = 28 bytes. More info about measure telegram later in this chapter
1F	Reading this register returns status and warnings, 2 bytes variable
21	Reading this register returns HF gain setting, 2 bytes variable
22	Reading this register returns pump driver voltage setting, 2 bytes variable
23	Reading this register returns pump current setting in constant current mode or output power setting in constant power mode, 2 bytes variable
24	Reading this register returns pump temperature setting, 2 bytes variable
25	Reading this register returns fiberlaser temperature setting in temperature tuning mode or wavelength setting in wavelength tuning mode, 2 bytes variable
28	Reading this register returns wavelength offset, 2 bytes variable
60	Reading this register returns Module address, 1 byte variable
61	Reading this register returns Module type, 1 byte variable. 21 hex = BasiK Module
62	Reading this register returns Hardware version, 2 bytes variable
64	Reading this register returns Firmware version, 2 bytes variable
65	Reading this register returns Module S/N, 8 bytes ascii
69	Reading this register returns Module error log, 12 bytes
6A	Reading this register returns Module measurement log, 12 x 2 bytes variable
6B	Reading this register returns Module current errors, 12 bytes
41	Reading this register returns fiberlaser measurement parameters, 16 bytes
42	Reading this register returns pump peltier current measurement parameters, 16 bytes
43	Reading this register returns fiberlaser petier current measurement parameters, 16 bytes
44	Reading this register returns pump temperature measurement parameters, 16 bytes
45	Reading this register returns pump current measurement parameters, 16 bytes
46	Reading this register returns pump monitor current measurement parameters, 16 bytes
47	Reading this register returns pump voltage measurement parameters, 16 bytes
48	Reading this register returns fiberlaser output power measurement parameters, 16 bytes
49	Reading this register returns module temperature measurement parameters, 16 bytes
4A	Reading this register returns pump driver voltage measurement parameters, 16 bytes
4B	Reading this register returns module input voltage measurement parameters, 16 bytes
4C	Reading this register returns wavelength calculation parameters, 16 bytes
51	Reading this register returns HF gain settings parameters, 16 bytes
52	Reading this register returns pump driver voltage settings parameters, 16 bytes
53	Reading this register returns pump current settings parameters, 16 bytes
54	Reading this register returns pump temperature settings parameters, 16 bytes
55	Reading this register returns fiberlaser temperature settings parameters, 16 bytes
56	Reading this register returns wavelength settings parameters, 16 bytes
57	Reading this register returns fiberlaser output power settings parameters, 16 bytes

Status register (1 byte)

Each byte in the status register indicates a state or a configuration of the BasiK module.

Bit	
0	0: Emission off, 1: Emission on
1	0: Constant Current mode, 1: Constant Power mode
2	0: Piezo tuning disabled, 1: Piezo tuning enabled
3	0: HF gain circuit disabled, 1: HF gain circuit enabled
4	0: Temperature tuning, 1: Wavelength tuning
5	0: Fiberlaser temperature not stable, 1: Fiberlaser temperature stable
6	0: Pump temperature not stable, 1: Pump temperature stable
7	0: General system enable HighZ, 1: General system enable dragged low by module

Warning register (1 byte)

Value	
1	A measurement has exceeded a non critical limit
2	A measurement has exceeded a critical limit, module is shut down
3	A measurement has exceeded a critical limit, module has shut down complete system

Measure telegram (28 bytes)

Byte	
1	Status register, 8 bit value
2	Warning register, 8 bit value
3,4	Fiberlaser temperature in 1/1000 °C, 16 bit unsigned value
5,6	Pump peltier current in mA, 16 bit unsigned value
7,8	Fiberlaser peltier current in mA, 16 bit signed value
9,10	Pump temperature in 1/1000 °C, 16 bit unsigned value
11,12	Pump current in mA, 16 bit unsigned value
13,14	Pump monitor current in µA, 16 bit unsigned value
15,16	Pump voltage in mV, 16 bit unsigned value
17,18	Fiberlaser output power in 1/100 mW, 16 bit unsigned value
19,20	Module temperature in 1/10 °C, 16 bit signed value
21,22	Pump driver voltage in mV, 16 bit unsigned value
23,24	Module input voltage in mV, 16 bit unsigned value
25,26	Wavelength in pm, 16 bit unsigned value (Note 1)
27,28	Wavelength offset in nm, 16 bit unsigned value (Note 1)

Note 1: The actual wavelength is calculated from the fiberlaser temperature. A fiberlaser with wavelength 1556.021 nm => Wavelength offset = 1550 (byte 27,28) and wavelength = 6021 (byte 25,26)

Parameter telegram (16 bytes)

Byte	
1	Type, 8 bit
2	Warning handler, 8 bit value
3,4	Start up, 16 bit unsigned value
5,6	Factory start up, 16 bit unsigned value
7,8	Upper limit, 16 bit unsigned value
9,10	Lower limit, 16 bit unsigned value
11,12	Correction X, 16 bit signed value
13,14	Correction Y, 16 bit signed value
15,16	Correction B, 16 bit signed value

Writing 1byte: [Destination][Source][05][Register][Data][CRC msb][CRC lsb]

Writing 2 bytes: [Destination][Source][05][Register][Data lsb][Data msb][CRC msb][CRC lsb]

Register (Hex)	Description
30	Data=00h : Emission off, Data=01h : Emission on
31	Data=00h : Constant Current mode, Data=01h : Constant Power mode
32	Data=00h : Piezo tuning disabled, Data=01h : Piezo tuning enabled
33	Data=00h : HF gain circuit disabled, Data=01h : HF gain circuit enabled
34	Data=00h : Temperature tuning, Data=01h : Wavelength tuning
21	HF gain setting, 2 bytes variable
22	Pump driver voltage setting, 2 bytes variable
23	Pump current setting in constant current mode or output power setting in constant power mode, 2 bytes variable.
24	Pump temperature setting, 2 bytes variable
25	Fiberlaser temperature setting in temperature tuning mode or wavelength setting in wavelength tuning mode, 2 bytes variable
68	Data=0100h: Restart module, Data=AA00h: Restore factory settings and resart module
69	Data=01h: Clear Error log