PR-33-S ETHERNET PROTOCOL SPECIFICATION

The main purpose of the Ethernet connection is to collect measurement data from the instrument. For this data acquisition, you'll need to have suitable software on your computer.

You can program a data acquisition program yourself following the specifications below. For examples and ready-made applications, please contact K-Patents.

COMMUNICATION PROTOCOL

The communications protocol is based on **UDP/IP** to **port 50023**. It is a client/server protocol, where the sensor is the server and thus only sends information when the client (i.e. your computer) requests it. The server should answer to all requests within 100 ms.

Request format

The client to server communication, i.e. the requests sent from your computer to the sensor, is in binary format. The request packets contain the following binary data (all integers are in the network order, MSB first):

- 32-bit integer: packet number
- 32-bit integer: request ID
- (any): request data (depends on the request)
- (any): fill-in data

Important: The maximum size of the message is 1472 octets (bytes).

The **packet number** is echoed back by the sensor, but not processed in any way. The packet numbers do not have to be sequential, any 32-bit value is valid.

The **request ID** is a 32-bit value that identifies the requested function, for example sensor information.

The **request data** consists of 0 to 1464 octets of additional data associated with the request.

The **fill-in data** can be used to increase the number of octets in a message. Any number of NULL characters (0x00) may be added to the end of the request as long as the total size of the message does not exceed the maximum of 1472 octets. This may be useful, for example, if the client implementation uses fixed-length packets.

Response format

The response data sent by the sensor is in ASCII format. With the exception of the packet number, the data is human-readable. The data structure is very simple:

- packet number (32-bit integer)
- zero or more lines of ASCII (text) keys and values associated with these keys (for example temperature key and process temperature in Celsius)

The **packet number** is echoed back without change. The client (software on computer) can use the packet number to check the response against the packet number of the request.

The **message text** consists of lines of text, each line a single key (of one word) and its value or values. The values are separated from the key by an equal sign (=) and multiple values are separated by a comma. White space (space or tabulator) is allowed anywhere except within a single value or key name.

If the response consists of a character string, it is enclosed in double quotes (").

For example all these are valid message text lines:

```
ok
temp = 23.45
headhum = 13.32
LEDcnt = 8341
ChemCurve = 1.234, 3.21, 0.00, 4.37,
1.11, 0.00002, 2.1345
StatusMessage = "Normal Operation"
```

Note: All the key identifiers are case-insensitive. However, K-Patents recommends that they are written as in this specification.

The server (sensor) may send the response keys in any order. It will send the mandatory keys (marked with an asterisk below) of the specific request, but it may omit any other keys. The server may also send keys that are not specified in this document, but the client (computer) may ignore them.

Request and response errors

When the server (sensor) detects an error, it responds with an error message. An error message can be caused for example by an unknown request or inability to collect data for the mandatory keys of a response.

REQUEST-RESPONSE PAIR SPECIFICATION

The list below describes the *query messages*, i.e. request-response pairs, used for data collection via Ethernet. **Those response keys that are always sent are preceded by an asterisk (*).**

NULL message

The null message is included in the query messages for debugging purposes as it can be used to check whether the server is listening. The message gives a high-level 'ping' functionality.

Request ID	`0x00000000`
Request data	`(none)`
Response keys	`IP` *IP address*
	`MAC` *Ethernet MAC address*

Protocol version

The version query is responded with a value representing the server (sensor) protocol version.

Request ID	`0x00000001`
Request data	`(none)`
Response key	`*Version` *integer, the server protocol version (currently 3)*

Sensor information

The sensor information query gives the basic information of the sensor.

Request ID	`0x00000003`
Request data	`0x00000000` *always zero*
Response keys	`*SensorSerial` *integer, sensor serial number*
	`*SProcSerial` *integer, sensor processor card serial number*
	`*SensorVersion` *integer, version number of the sensor software*

Measurement results

The measurement result query gives the measured and calculated measurement values from the chosen sensor.

Request ID	`0x00000004`
Request data	`0x00000000` *always zero*
Response keys	`Status` *string, sensor status message*
	`PTraw` *integer, PT1000 value*
	`LED` *float, sensor led value*
	`RHsens` *float, sensor internal humidity*
	`nD` *float, calculated n_D value*
	`CONC` *float, final concentration value*
	`Tsens` *float, sensor internal temperature*
	`T` *float, process temperature*
	`CCD` *float, image shadow edge*
	`CALC` *float, calculated concentration value*
	`QF` *float, quality factor*
	`BGlight` *integer, background light*

ERROR MESSAGE SPECIFICATION

If the server (sensor) does not recognize the request or cannot fulfill it, it responds with an error message. The error message has the following keys:

`*Error`	integer	error code `0x00000001` *Unknown request*
`*Error`	integer	error code `0x00000002` *Invalid request (request recognized, invalid request data)*
`ErrorMsg`	string	*error details*

There may also be error-dependent extra keys.