# PENKO Engineering B.V.

Your Partner for Fully Engineered Factory Solutions


## Protocol description:
## PENKO PDI

PENKO
an ETC Company

# 1 Table of Contents

Introduction ..... 3
2 TP serial ..... 4
3 TP UDP ..... 6
4 TP data description ..... 7
4.1 Reply code 0x53: Device busy internal ..... 9
4.2 Reply code 0x54: Function parameter error ..... 9
4.3 Reply code 0x55: Function accepted and done ..... 9
4.4 Reply code 0x57: Host functions disabled ..... 9
4.5 Reply code 0x58: Internal status conflict ..... 9
4.6 Reply code 0x59: Unknown command ..... 9
5 PDI functions ..... 10
5.1 Check if the PDI feature is available ..... 11
5.2 Retrieve tree structure information ..... 11
5.3 Get property record ..... 13
5.4 Read property ..... 17
5.5 Write property ..... 19
5.6 Write property extended ..... 22

# PENKO PDI Protocol

## Introduction

The PENKO TP PDI protocol is a part of the PENKO Two Phase (TP) protocol. PDI stands for PENKO Device Interface and is used for communication between PENKO devices, PC's, PLC's and other PENKO equipment. The protocol can be used over serial (RS232, RS422, RS485, USB) and Ethernet (UDP) connections.

PDI is available in the current line of PENKO devices and shows the complete device configuration in a tree structure. Every property in the tree has a unique path number, and through these path numbers every property can be accessed.

The TP protocol is based on two parts; a request and a reply. Both phases use the same shape for sending data. The difference is made by who takes the initiative for the phase. In the first phase of the protocol, the request, the initiative is taken by the master. Most of the times this will be a PC application, but it can also be an application on a PLC or other embedded device. The second phase is initiated by the slave. This is the reply message. This role will be mostly fulfilled by an embedded device such as an indicator or a remote I/O unit. After sending the reply, the communication cycle is closed and a new request can be sent.

The PDI protocol is used in these PENKO software applications:

- *PENKO Pi Mach II Manage*
- *PENKO PDI Client*

With these applications the path numbers of all device properties can be found.

## 2 TP serial

The TP serial frames are constructed as follows.

**Request frame:**

| Byte | Byte | Byte | Byte[] | Byte | Byte | Byte |
|---|---|---|---|---|---|---|
| **DLE** | **STX** | **Address** | **Data** | **Checksum** | **DLE** | **ETX** |

**Reply frame:**

| Byte | Byte | Byte | Byte[] | Byte | Byte | Byte |
|---|---|---|---|---|---|---|
| **DLE** | **STX** | **Address** | **Data** | **Checksum** | **DLE** | **ETX** |

**Frame description:**

| Frame part | Description |
|---|---|
| **DLE + STX** | Preamble |
| **Address** | Port address set in device. For USB connection the address is always 0. |
| **Data** | Command code, Operation code and parameters, described in chapter Data description |
| **Checksum** | Checksum, described in Checksum calculation |
| **DLE + ETX** | Postamble |

**Used characters:**

| Character | Decimal | Hexadecimal | Description |
|---|---|---|---|
| **STX** | 2 | 0x02 | Start of TeXt |
| **ETX** | 3 | 0x03 | End of TeXt |
| **DLE** | 16 | 0x10 | Data Link Escape |

The postamble must be the only DLE+ETX sequence in the protocol in order to indicate the end of a frame. For every character in the address, data or checksum that equals the DLE character, an extra DLE character is added to ensure the uniqueness of the postamble. The extra DLE characters are not included in the checksum calculation.

**Checksum calculation:**

The checksum is the inverted first complement of the sum of the address byte and all data bytes. The possible extra DLE characters, as described above, are not included in this calculation.

**Example:**

The sum of the address byte and data bytes (excluding possible extra DLE characters) is 0x1234

Get the first complement: 0x1234 ***AND*** 0xFF = 0x34

Inverse the result: 0x34 ***XOR*** 0xFF = 0xCB

# 3 TP UDP

The TP UDP frames are constructed as follows.

**Request frame:**

| Byte | Byte | Byte | Byte | Byte[] |
|---|---|---|---|---|
| **0x00** | **0x00** | **0x00** | **0x00** | **Data** |

**Reply frame:**

| Byte | Byte | Byte | Byte | Byte[] |
|---|---|---|---|---|
| **0x00** | **0x00** | **0x00** | **0x00** | **Data** |

**Frame description:**

| Frame part | Description |
|---|---|
| **0x00** | Preamble, default 4 x 0x00, reserved for future extensions |
| **Data** | Command code, Operation code and parameters, described in chapter Data description |

Compared to the serial TP frame, the TP protocol over Ethernet has no address, preamble, postamble, checksum and extra DLE characters in the communication frame.

# 4 TP data description

The data frames in the serial and UDP communication are constructed as follows.

**Request data frame:**

| Byte | Byte | Byte[] |
|---|---|---|
| **Command code** | **Operation code** | **Parameters** |

**Reply data frame:**

| Byte | Byte | Byte[] |
|---|---|---|
| **Command code** | **Operation code** | **Parameters** |

**Frame description:**

| Frame part | Description |
|---|---|
| **Command code** | Feature of the device to perform the request on |
| **Operation code** | Operation to perform on the feature, not present in all features |
| **Parameters** | Parameters belonging to the operation, not present in all operation |

**A complete TP serial frame:**

| Byte | Byte | Byte | Byte[] | Byte | Byte | Byte |
|---|---|---|---|---|---|---|
| **DLE** | **STX** | **Address** | **Data** | **Checksum** | **DLE** | **ETX** |

| Byte | Byte | Byte[] |
|---|---|---|
| **Command code** | **Operation code** | **Parameters** |

**A complete TP UDP frame:**

| Byte | Byte | Byte | Byte | Byte[] |
|---|---|---|---|---|
| **0x00** | **0x00** | **0x00** | **0x00** | **Data** |

| Byte | Byte | Byte[] |
|---|---|---|
| **Command code** | **Operation code** | **Parameters** |

The following command code is used for PDI. Usually the command code in the request frame is replied in the reply frame.

**PDI command code (request and reply):**

| Command code | Name | Description |
|---|---|---|
| **0xB4** | PDI | PDI functions |

In some cases only an acknowledge command code is returned, or in case a reply cannot be sent, one of the error command codes is returned.

**Reply codes:**

| Command code | Name | Description |
|---|---|---|
| **0x53** | BUSY | Device busy internal |
| **0x54** | ERROR | Function parameter error |
| **0x55** | ACK | Function accepted and done |
| **0x57** | DISABLED | Host functions disabled |
| **0x58** | NAK | Internal status conflict |
| **0x59** | ILLEGAL | Unknown command |

## 4.1 Reply code 0x53: Device busy internal

When a function is applied while the slave device has been long engaged in another activity, e.g. user input, this result code can be returned. When receiving this result the master can decide if he continues to poll the relevant slave until it can answer or that another device is accessed. This result code is only applicable to single-threaded applications, which often have to do with user input.

## 4.2 Reply code 0x54: Function parameter error

Upon receiving a function call, a check is done on the number of bytes received. When the number of bytes does not match the number specified for the requested feature, this feature is not implemented and this code is returned. This error can be caused by a transmission error in which a character is lost but the checksum is still correct. More likely it is that a mistake was made with building the request packet that a wrong packet size is specified.

## 4.3 Reply code 0x55: Function accepted and done

A function to activate an action on the slave device without returning data will give this result code to the master after the execution of the action.

## 4.4 Reply code 0x57: Host functions disabled

Slave devices with a configurable user interface can disable the protocol driver so remote configuration and / or control is no longer possible. In this case the protocol driver delivers this result code back. This makes it possible for the master to give a detailed error notification to the user and possibly remove the device from the communication.

## 4.5 Reply code 0x58: Internal status conflict

Performing a function can be connected to the internal status of the slave device. If this is the case, this function result code comes back. Such a situation may occur when, e.g., the parameters of a slave process are changed while the process is active. It is therefore prudent to keep track of the status of the slave device on the master so this conflict can be prevented.

## 4.6 Reply code 0x59: Unknown command

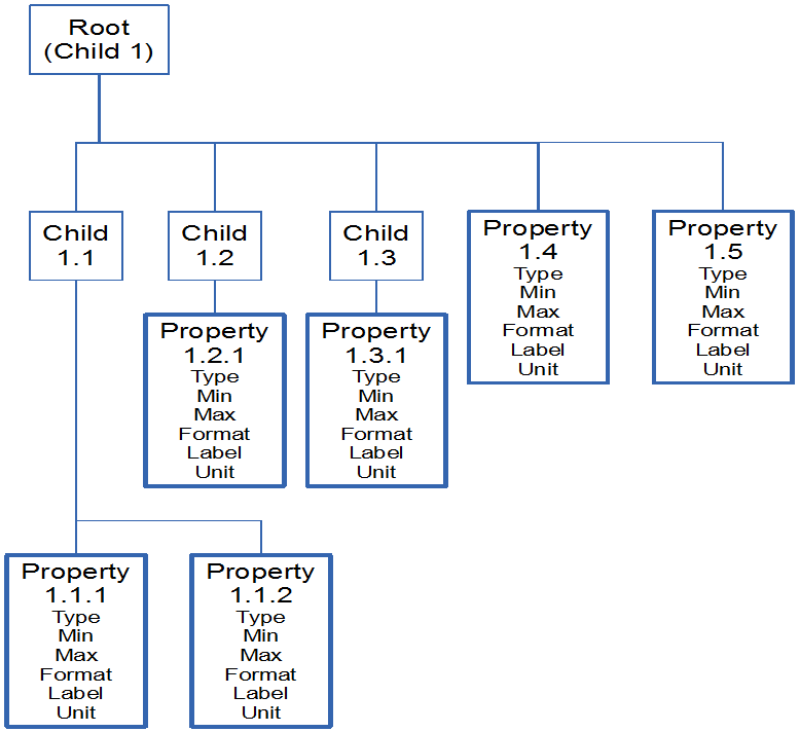
When a function is applied which is not known to the slave device, this result code is returned. By asking for the device ID there can be determined what kind of device is hidden behind the device address. On this basis the commands that are valid for this application can be determined. Possibly you may need to request additional information such as the version number from the slave device.

# 5 PDI functions

With this interface all settings of the device can be viewed and edited. The device takes care of the validation of the data so it's not possible to enter invalid settings. Access to legally relevant settings always go through the WELMEC library, so adjustments to the measurement system are not readily possible and are always registered.

The PDI data forms a tree structure with the device itself as the highest level. A multi-level data under there is a 'Child'. Each 'Child' has some properties or another node. Property or Child is selected with an address. The first child is 1. The first Child or Property under there is 1.1 and thus there's added another number for each layer. Up to 256 nodes at 256 levels can be approached.

From level 1 there can be retrieved how many properties and children there are at this level. Thereafter, the properties (data type, name, size, and unit) in each to be set value are retrieved with PDI_PROPERTY.



| OpCode | Description |
|---|---|
| **0** | Feature detection |
| **1** | Retrieve tree structure information |
| **2** | Get property record |
| **3** | Read property |
| **4** | Write property |
| **5** | Write property extended |

## 5.1 Check if the PDI feature is available

The availability of the PDI interface can be detected with this function. If present, a reply with the system command ACK follows.

Request:

| Command | OpCode |
|---|---|
| 0xB4 | 0x00 |

Reply (ACK):

| Command |
|---|
| 0x55 |

## 5.2 Retrieve tree structure information

The number of child items and properties of the selected node and the name of the node are returned with this command. Strings are null terminated to indicate the end of the string.

Request:

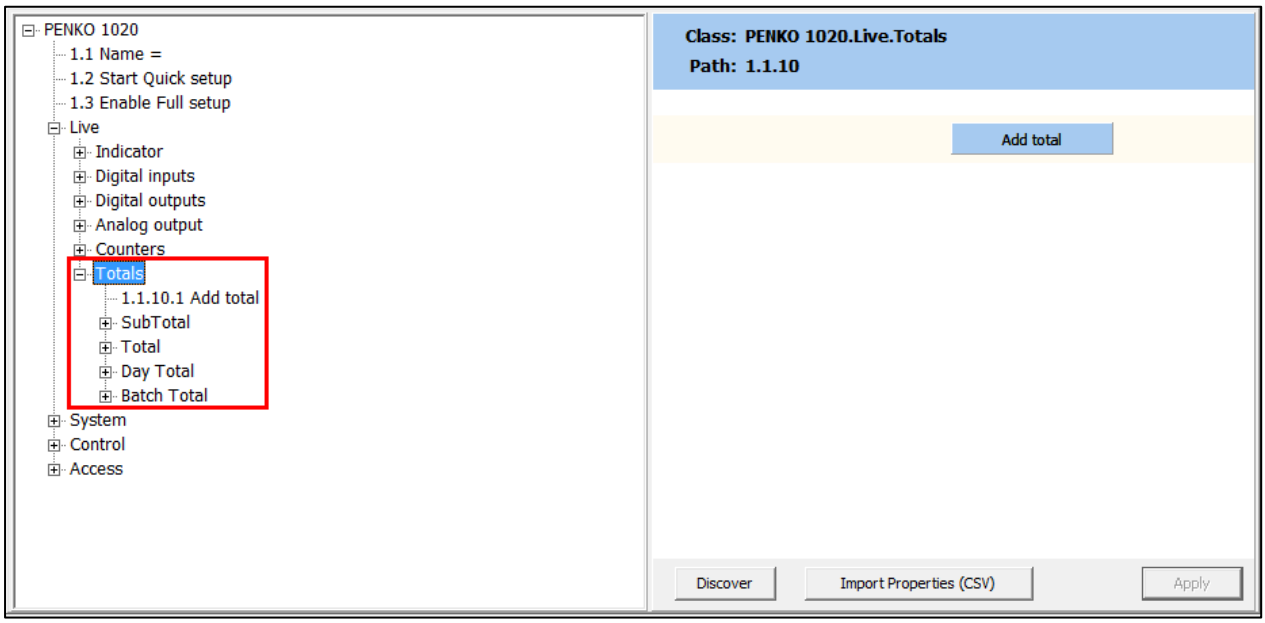
| Command | OpCode | Node |
|---|---|---|
| 0xB4 | 0x01 | Byte [] |

Reply:

| Command | OpCode | Node | # of children | # of properties | Node name |
|---|---|---|---|---|---|
| 0xB4 | 0x01 | Byte [] | 0x00 | 0x00 | Byte [] |

**Examples**

The live totals node in the PENKO 1020 is found in Live, path number 1.1.10.




To enumerate this node send the following frame (node 1.1.10 = 0x01010A):

Request:

| Command | OpCode | Node |
|---|---|---|
| 0xB4 | 0x01 | 0x01010A |

Reply:

| Command | OpCode | Node | # of children | # of properties | Node name |
|---|---|---|---|---|---|
| 0xB4 | 0x01 | 0x01010A | 0x04 | 0x01 | 0x54 6F 74 61 6C 73 00 |

- The number of children is 4
- The number of properties is 1
- The name of the node is (HEX to ASCII) Totals, terminated with a NULL

## 5.3 Get property record

This indicates the characteristics of the selected node, including type, minimum value, maximum value, number format (number of digits and point position), label name and the unit. The unit field may also include a series of texts to give an enumeration as a list of options. When the selected node is a child node then only the label is of interest. Strings are null terminated to indicate the end of the string. Empty strings only have a terminator.

For the **attribute** and **format** bits; when more than one value is applicable, the values are added together.

Request:

| Command | OpCode | Node | Property index |
|---|---|---|---|
| 0xB4 | 0x02 | Byte [] | 0x00 |

Reply:

| Command | OpCode | Node | Property index | Record type | Min value | Max value |
|---|---|---|---|---|---|---|
| 0xB4 | 0x02 | Byte [] | 0x00 | 0x00 | 0x00000000 | 0x00000000 |

| Attribute | Format | Label | Unit/enumeration |
|---|---|---|---|
| 0x0000 | 0x0000 | Byte [] | Byte [] |

**Record type and attribute bits:**

| Record type | Description |
|---|---|
| **0x00** | Invalid |
| **0x01** | Standard |
| **0x02** | Enumeration |

| Attribute bits | Description |
|---|---|
| **0x0001** | Read |
| **0x0002** | Write |
| **0x0010** | Button |
| **0x0020** | Inform user |
| **0x1000** | Rebuild |
| **0x2000** | Live |
| **0x4000** | Update parent |
| **0x8000** | Update root |

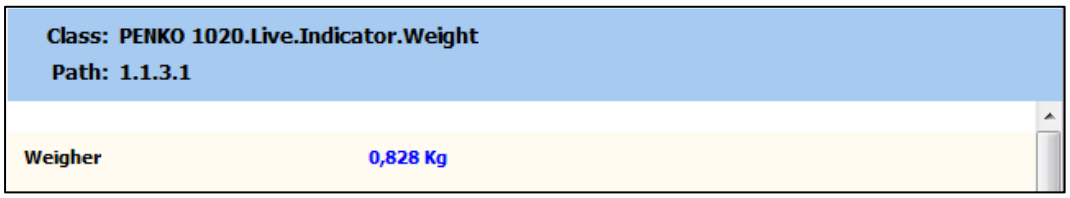
**Format bits:**

| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 | Description |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | Signed |
| 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | Unsigned |
| - | 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | Zero suppressing |
| - | 0 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | No zero suppressing |
| - | - | 0 | 0 | - | - | - | - | 0 | - | - | - | 0 | - | - | - | Type: Numeric |
| - | - | 0 | 0 | - | - | - | - | 0 | - | - | - | 1 | - | - | - | Type: Float |
| - | - | 0 | 0 | - | - | - | - | 1 | - | - | - | 0 | - | - | - | Type: Ulong |
| - | - | 0 | 0 | - | - | - | - | 1 | - | - | - | 1 | - | - | - | Type: Hex |
| - | - | 0 | 1 | - | - | - | - | 0 | - | - | - | 0 | - | - | - | Type: Time |
| - | - | 0 | 1 | - | - | - | - | 0 | - | - | - | 1 | - | - | - | Type: String |
| - | - | 0 | 1 | - | - | - | - | 1 | - | - | - | 0 | - | - | - | Type: Spin |
| - | - | 0 | 1 | - | - | - | - | 1 | - | - | - | 1 | - | - | - | Type: Labeled |
| - | - | 1 | 0 | - | - | - | - | 0 | - | - | - | 0 | - | - | - | Type: Date |
| - | - | 1 | 0 | - | - | - | - | 0 | - | - | - | 1 | - | - | - | Type: Password |
| - | - | 1 | 0 | - | - | - | - | 1 | - | - | - | 1 | - | - | - | Type: Weight |
| - | - | 1 | 1 | - | - | - | - | 0 | - | - | - | 0 | - | - | - | Type: IP address |
| - | - | - | - | 0 | 0 | 0 | 0 | - | - | - | - | - | - | - | - | Step size 1 |
| - | - | - | - | 0 | 0 | 0 | 1 | - | - | - | - | - | - | - | - | Step size 2 |
| - | - | - | - | 0 | 0 | 1 | 0 | - | - | - | - | - | - | - | - | Step size 5 |
| - | - | - | - | 0 | 0 | 1 | 1 | - | - | - | - | - | - | - | - | Step size 10 |
| - | - | - | - | 0 | 1 | 0 | 0 | - | - | - | - | - | - | - | - | Step size 20 |
| - | - | - | - | 0 | 1 | 0 | 1 | - | - | - | - | - | - | - | - | Step size 50 |
| - | - | - | - | 0 | 1 | 1 | 0 | - | - | - | - | - | - | - | - | Step size 100 |
| - | - | - | - | 0 | 1 | 1 | 1 | - | - | - | - | - | - | - | - | Step size 200 |
| - | - | - | - | 1 | 0 | 0 | 0 | - | - | - | - | - | - | - | - | Step size 500 |
| - | - | - | - | 1 | 0 | 0 | 1 | - | - | - | - | - | - | - | - | Step size 1000 |
| - | - | - | - | 1 | 0 | 1 | 0 | - | - | - | - | - | - | - | - | Step size 2000 |
| - | - | - | - | 1 | 0 | 1 | 1 | - | - | - | - | - | - | - | - | Step size 5000 |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 0 | No decimal point |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 1 | 1 decimal position |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 0 | 2 decimal positions |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 1 | 3 decimal positions |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 0 | 4 decimal positions |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 0 | 1 | 5 decimal positions |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 0 | 6 decimal positions |
| - | - | - | - | - | - | - | - | - | - | - | - | - | 1 | 1 | 1 | Auto decimal positions |

**Examples**

***Get standard record:***

The live weigher value of the PENKO 1020 is found in Live - Indicator - Weight, path number 1.1.3.1 property 1.




To gain its properties, the following request is sent:

Request:

| Command | OpCode | Node | Property index |
|---|---|---|---|
| 0xB4 | 0x02 | 0x01010301 | 0x01 |

Reply:

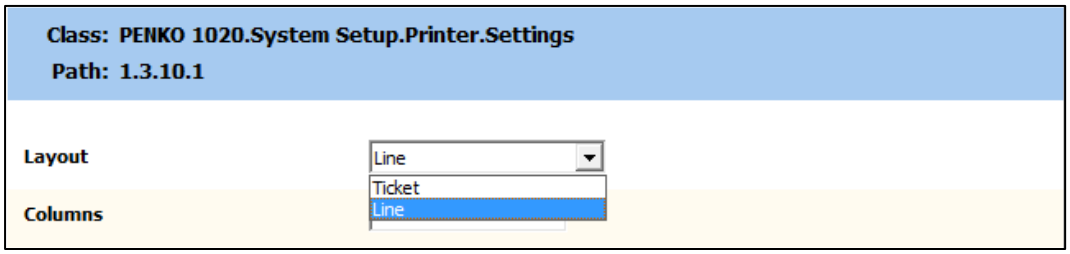
| Command | OpCode | Node | Property index | Record type | Min value |
|---|---|---|---|---|---|
| 0xB4 | 0x02 | 0x01010301 | 0x01 | 0x01 | 0x00000000 |

| Max value | Attribute | Format | Label | Unit/enumeration |
|---|---|---|---|---|
| 0x00000000 | 0x2001 | 0xC003 | 0x5765696768657200 | 0x4B6700 |

- The record type is 1, so it's a standard record
- The min and max value are both 0 so not applicable for this property
- The attribute is 0x2001 → 0x2000 = **Live** + 0x0001 = **Read** → read only live property
- The format is 0xC003 → 11000000 00000011
  - Signed
  - Zero suppressing
  - Type: Numeric
  - Step size 1
  - 3 decimal positions
- The label is (HEX to ASCII) Weigher, terminated with a NULL
- The unit is (HEX to ASCII) Kg, terminated with a NULL

***Get enumerated record:***

The printer layout setting of the PENKO 1020 is found in System Setup - Printer - Settings, path number 1.3.10.1, property 1.




To gain its properties, the following request is sent:

Request:

| Command | OpCode | Node | Property index |
|---|---|---|---|
| 0xB4 | 0x02 | 0x01030A01 | 0x01 |

Reply:

| Command | OpCode | Node | Property index | Record type | Min value | Max value |
|---|---|---|---|---|---|---|
| 0xB4 | 0x02 | 0x01030A01 | 0x01 | 0x02 | 0x00000000 | 0x00000001 |

| Attribute | Format | Label | Unit/enumeration |
|---|---|---|---|
| 0x0003 | 0x1080 | 0x4C61796F757400 | 0x5469636B6574004C696E6500 |

- The record type is 2, so it's an enumerated record
- The min value is 0
- The max value is 1 so there are 2 records, record 0 and record 1
- The attribute is 0x003 → 0x0001 = **Read** + 0x0002 = **Write** → read/write property
- The format is 0x1080 → 00010000 10000000
    - Unsigned (not applicable)
    - No zero suppressing (not applicable)
    - Type: Spin
    - Step size 1 (not applicable)
    - No decimal point (not applicable)
- The label is (HEX to ASCII) Layout, terminated with a NULL
- The first enumeration is (HEX to ASCII) Ticket, terminated with a NULL
- The second enumeration is (HEX to ASCII) Line, terminated with a NULL

## 5.4 Read property

The value of the selected property is returned with this command. Strings are null terminated to indicate the end of the string. Empty strings only have a terminator.

The reply contains a status byte to indicate success or error:

| Status byte | Description |
|---|---|
| **0x00** | Error |
| **0x01** | OK |

Request:

| Command | OpCode | Node | Property index |
|---|---|---|---|
| 0xB4 | 0x03 | Byte [] | 0x00 |

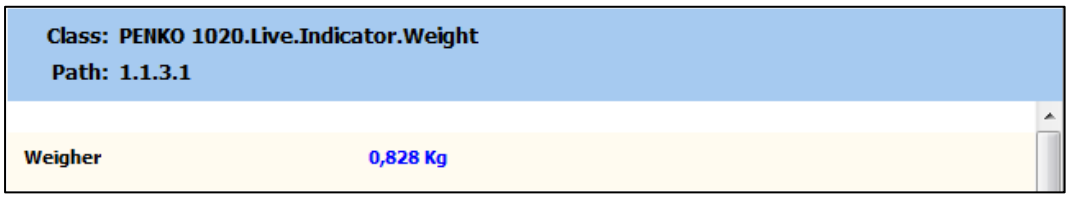
Reply:

| Command | OpCode | Node | Property index | Status | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x03 | Byte [] | 0x00 | 0x00 | Byte [] |

**Examples**

***Read weight value:***

The live weigher value of the PENKO 1020 is found in Live - Indicator - Weight, path number 1.1.3.1 property 1.




To gain its properties, the following request is sent:

Request:

| Command | OpCode | Node | Property index |
|---|---|---|---|
| 0xB4 | 0x03 | 0x01010301 | 0x01 |

Reply:

| Command | OpCode | Node | Property index | Status | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x03 | 0x01010301 | 0x01 | 0x01 | 0x0000033C |

- The status is OK (status 0x01)
- The weight is 0x0000033C → 828

***Tare active indication:***

The live tare active indication of the PENKO 1020 is found in Live - Status, path number 1.1.3.2 property 9.

Request:

| Command | OpCode | Node | Property index |
|---|---|---|---|
| 0xB4 | 0x03 | 0x01010302 | 0x09 |

Reply:

| Command | OpCode | Node | Property index | Status | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x03 | 0x01010302 | 0x09 | 0x01 | 0x00000001 |

- The status is OK (status 0x01)
- The tare active indication is 0x00000001 → tare is active

## 5.5 Write property

This command specifies a new value for the selected property. When this value is set in the device as read-only it will not be changed. Strings are null terminated to indicate the end of the string. Empty strings only have a terminator.

The reply contains a “save ok” byte to indicate success or error:

| Save OK | Description |
|---|---|
| **0x00** | Save failed |
| **0x01** | Save successful |
| **0x02** | Save none *(command executed successfully but no value had to be saved)* |

Request:

| Command | OpCode | Node | Property index | Terminator | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x04 | Byte [] | 0x00 | 0x00 | Byte []* |

* *in case of a value, use long format, 0x00 00 00 00*

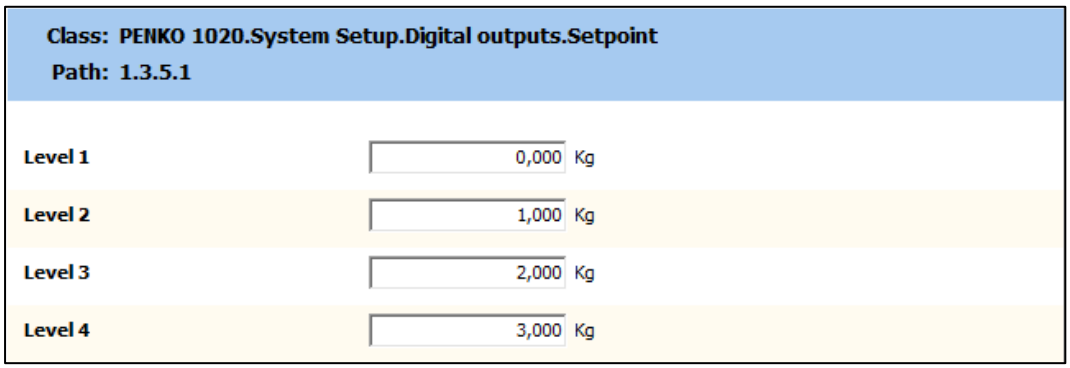
Reply:

| Command | OpCode | Node | Property index | Terminator | Property value | Save OK |
|---|---|---|---|---|---|---|
| 0xB4 | 0x04 | Byte [] | 0x00 | 0x00 | Byte [] | 0x00 |

**Examples**

***Change setpoint:***

The setpoint of digital output 1 of the PENKO 1020 is found in System Setup - Digital outputs - Setpoint, path number 1.3.5.1 property 1.




To set this setpoint to 300 (hex 0x0000012C), the following request is sent:

Request:

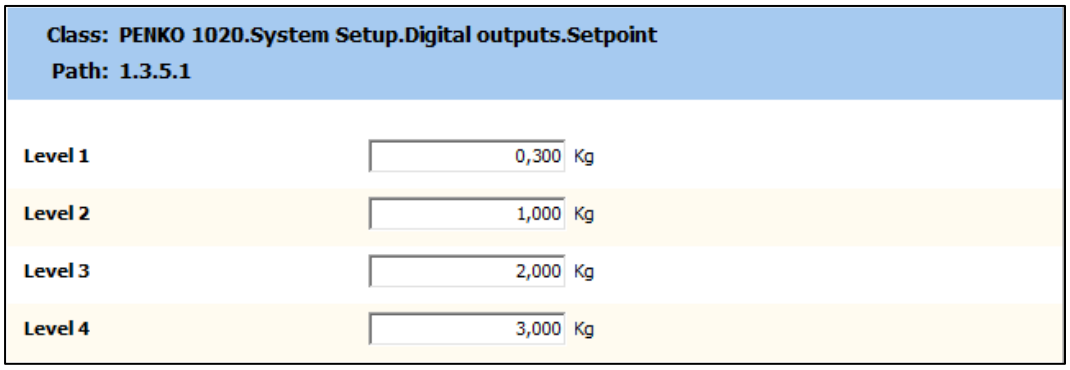
| Command | OpCode | Node | Property index | Terminator | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x04 | 0x01030501 | 0x01 | 0x00 | 0x0000012C |

Reply:

| Command | OpCode | Node | Property index | Terminator | Property value | Save OK |
|---|---|---|---|---|---|---|
| 0xB4 | 0x04 | 0x01030501 | 0x01 | 0x00 | 0x0000012C | 0x01 |

- The set value is 0x0000012C → decimal 300
- Save is succeeded (save ok 0x01)

Result:



***Set/reset zero:***

The zero control of the PENKO 1020 is found in Control - Indicator - Zero, path number 1.6.1.1 property 1 and 2.

To set the indicator to zero, the following request is sent:

Request:

| Command | OpCode | Node | Property index | Terminator | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x04 | 0x01060101 | 0x01 | 0x00 | 0x00000000* |

* *value is not applicable for this command but has to be entered to complete the frame*

Reply:

| Command | OpCode | Node | Property index | Terminator | Property value | Save OK |
|---|---|---|---|---|---|---|
| 0xB4 | 0x04 | 0x01 06 01 01 | 0x01 | 0x00 | 0x00 00 00 00 | 0x02 |

- Save is none (save ok 0x02) meaning there was no value to save but the command is executed successfully

Result:

| | |
|---|---|
| **Weigher** | **0,000 Kg** |

To reset zero, property 2 is used:

Request:

| Command | OpCode | Node | Property index | Terminator | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x04 | 0x01060101 | 0x02 | 0x00 | 0x00000000* |

* *value is not applicable for this command but has to be entered to complete the frame*

The reply will be as follows:

| Command | OpCode | Node | Property index | Terminator | Property value | Save OK |
|---|---|---|---|---|---|---|
| 0xB4 | 0x04 | 0x01060101 | 0x02 | 0x00 | 0x00000000 | 0x02 |

- Save is none (save ok 0x02) meaning there was no value to save but the command is executed successfully

Result:

| | |
|---|---|
| **Weigher** | **0,128 Kg** |

## 5.6 Write property extended

This command is equal to the write property command, only now an extra string is returned in the reply frame. In case a command succeeded, only a NULL is returned as string. In case an error occurs, the error string is returned. Strings are null terminated to indicate the end of the string. Empty strings only have a terminator.

The reply contains a “save ok” byte to indicate success or error:

| Save OK | Description |
|---|---|
| **0x00** | Save failed |
| **0x01** | Save successful |
| **0x02** | Save none *(command executed successfully but no value had to be saved)* |

Request:

| Command | OpCode | Node | Property index | Terminator | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x05 | Byte [] | 0x00 | 0x00 | Byte []* |

** in case of a value, use long format, 0x00 00 00 00*

Reply:

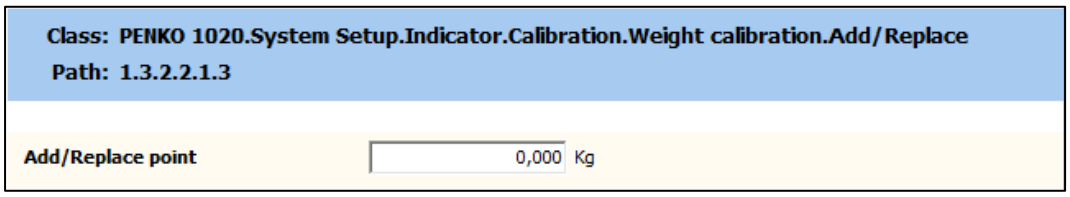
| Command | OpCode | Node | Property index | Terminator | Property value | Save OK | Reply string |
|---|---|---|---|---|---|---|---|
| 0xB4 | 0x05 | Byte [] | 0x00 | 0x00 | Byte [] | 0x00 | Byte []* |

****** NULL when succeeded, error string when error occurs*

**Examples**

***Set calibration point with succeed reply:***

The add calibration point of the PENKO 1020 is found in System Setup - Indicator - Calibration - Add/Replace, path number 1.3.2.2.1.3 property 1.




To set the zero point, the following request is sent:

Request:

| Command | OpCode | Node | Property index | Terminator | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x05 | 0x010302020103 | 0x01 | 0x00 | 0x00000000 |

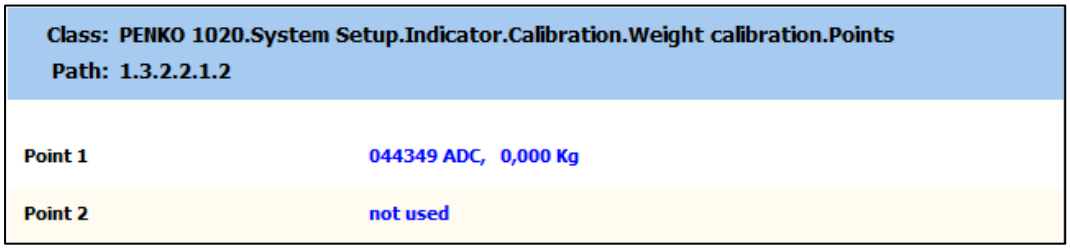
Reply:

| Command | OpCode | Node | Property index | Terminator | Property value | Save OK |
|---|---|---|---|---|---|---|
| 0xB4 | 0x05 | 0x010302020103 | 0x01 | 0x00 | 0x00000000 | 0x01 |

| Reply string |
|---|
| 0x00 |

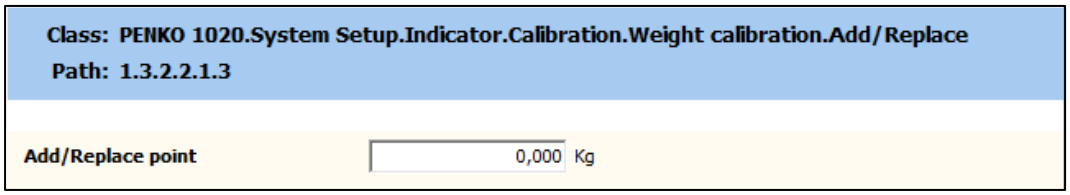
- The set value is 0x00000000 → decimal 0
- Save is succeeded (save ok 0x01)
- The reply string is NULL meaning succeeded

Result:



***Set calibration point with error reply:***

The add calibration point of the PENKO 1020 is found in System Setup - Indicator - Calibration - Add/Replace, path number 1.3.2.2.1.3 property 1.




To set the gain point to 100.000 (*too high, will cause error*), the following request is sent:

Request:

| Command | OpCode | Node | Property index | Terminator | Property value |
|---|---|---|---|---|---|
| 0xB4 | 0x05 | 0x010302020103 | 0x01 | 0x00 | 0x000186A0 |

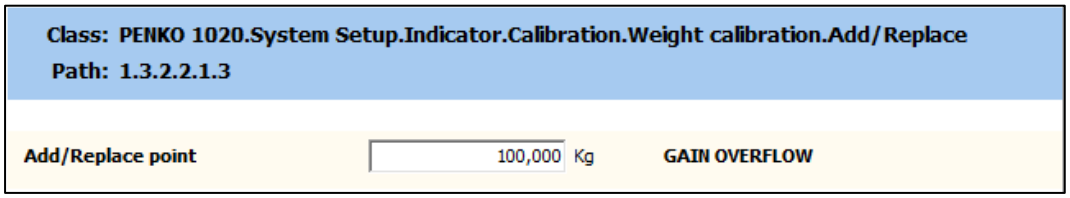
Reply:

| Command | OpCode | Node | Property index | Terminator | Property value | Save OK |
|---|---|---|---|---|---|---|
| 0xB4 | 0x05 | 0x010302020103 | 0x01 | 0x00 | 0x000186A0 | 0x00 |

| Reply string |
|---|
| 0x47 41 49 4E 20 4F 56 45 52 46 4C 4F 57 00 |

- The set value is 0x000186A0 → decimal 100.000
- Save is failed (save ok 0x00)
- The reply string is (HEX to ASCII) GAIN OVERFLOW, terminated with a NULL

Result:




**About PENKO**

At PENKO Engineering we specialize in weighing. Weighing is inherently chemically correct, independent of consistency, type or temperature of the raw material. This means that weighing any kind of material guaranties consistency and thus, it is essential to sustainable revenue generation in any industry. As a well-established and proven solution provider, we strive for the ultimate satisfaction of custom design and/or standard applications, increasing your efficiencies and saving you time, saving you money.

Whether we are weighing raw materials, components in batching, ingredients for mixing or dosing processes, - or weighing of static containers and silos, or - in-motion weighing of railway wagons or trucks, by whatever means required during a process, we are essentially forming vital linkages between processes and businesses, anywhere at any time. We design, develop and manufacture state of the art technologically advanced systems in accordance with your strategy and vision. From the initial design brief, we take a fresh approach and a holistic view of every project, managing, supporting and/or implementing your system every step of the way. Curious to know how we do it? www.penko.com

**Certifications**

PENKO sets high standards for its products and product performance which are tested, certified and approved by independent expert and government organizations to ensure they meet – and even – exceed metrology industry guidelines. A library of testing certificates is available for reference on:
**http://penko.com/nl/publications_certificates.html**

**PENKO Professional Services**

PENKO is committed to ensuring every system is installed, tested, programmed, commissioned and operational to client specifications. Our engineers, at our weighing center in Ede, Netherlands, as well as our distributors around the world, strive to solve most weighing-system issues within the same day. On a monthly basis PENKO offers free training classes to anyone interested in exploring modern, high-speed weighing instruments andsolutions. Training sessions on request: **www.penko.com/training**

**PENKO Alliances**

PENKO's worldwide network: Australia, Brazil, China, Denmark, Germany, Egypt, Finland, France, India, Italy, Netherlands, Norway, Poland, Portugal, Slovakia, Spain, Syria, Turkey, United Kingdom, South Africa, Slovakia Sweden and Switzerland, Singapore.
A complete overview you will find on: **www.penko.com/dealers**

**PENKO Engineering B.V.** ▪ Schutterweg 35, NL 6718XC Ede ▪ Tel +31 (0) 318525630 ▪ Fax +31 (0) 31852971 ▪ info@penko.com
**Web ▪ www.penko.com** ▪  7600L1103 EN PENKO PDI Protocol R.4