Technical Documentation

QR code to the website
Technical Documentation

FAFNIR Universal Device Protocol

Art. No.	Version	Edition
350052	**1.10**	**2025-09**

FAFNIR GmbH • Schnackenburgallee 149 c • 22525 Hamburg • Germany • Tel.: +49 / 40 / 39 82 07–0 • Fax: +49 / 40 / 390 63 39

FAFNIR

Table of contents

1 Interface

Modulation:	RS-485
Signals:	A, B, GND
Mode:	Asynchronous, half duplex
Baud rate:	1200 bps / 4800 bps
Start-bits:	1
Data-bits:	8
Stop-bits:	1
Parity:	-

Timing conditions:

The time to transmit one character depends on the baud rate (4800 bps: 2 ms, 1200 bps: 8 ms). Since the processing time for the answer or the next command can be assumed to be the same for both speeds, timeouts are doubled for 1200 bps and set into brackets.

If the first character of the response is not received within 50 ms (100 ms) or more, the next command can be sent. The transmission should be continuous. The inter-character delay must be shorter than 20 ms (40 ms).

The host system must be able to switch off the RS-485 driver within 10 ms (20 ms) after the command was finished. The host interface must set the RS-485 to a definite state when the driver is switched off.

Note: Please read the Technical Documentation of the devices using this protocol to check if there is any kind of limitation (e.g. supported baud rate).

2 Abbreviations and formats

F - If 1st character: Header character for dynamic data read request & response.

G - If 1st character: Header character for static data read request & response.

X - If 1st character: Header character for static data write request & response.

Y - If 1st character: Header character for dynamic data write request & response.

AC - Multiplexer (e.g. VPI) board address and channel, 8-bit hex value, 00H...FFH, ASCII coded, always 2 characters. Hex value letter characters must be in uppercase format, i.e. 'A'...'F'.
Bit 7...3 - Multiplexer board address (0 = board address 1, 31 = board address 32)
Bit 2...0 - Multiplexer channel (0 = channel 1, 7 = channel 8)

D - Device type (a...w), always 1 lower case character, ASCII coded.

SN - Decimal positive integer value with variable field width, ASCII coded, representing the serial number of a device. In a message dialogue, the ID '#' precedes the SN, making up the "serial number data field": <# SN>.

Note: if the "serial number data field" is required, it must directly follow the "Device type" (D) character.

N...N	-	Decimal integer value with variable field width, ASCII coded, all negative values have a ‘-‘ as prefix, all positive values (including 0) have no prefix. A value of -0 indicates that due to an error this value is currently not available.
X...X	-	Hex value with variable field width, ASCII coded. Hex value letter characters must be in uppercase format, i.e. 'A'...'F'. A value of "-0" indicates that due to an error this value is currently not available.
cs	-	Checksum To be able to detect messages that have been corrupted during the transmission a CRC16 checksum is included in the request and response messages. The polynomial that is used for the CRC calculation is of type CRC_CCITT ($X^{16} + X^{12} + X^{5} + 1$). The seed (start value) of the CRC calculation is 0. CRC calculation starts with the first character of the message and ends with (and includes) the character directly in front of the first checksum character. In the request message only the lower byte of the 2-byte checksum is transmitted while in the response message the complete checksum is included. The checksum is ASCII coded. Hex value letter characters must be in uppercase format, i.e. 'A'...'F'.
cr	-	Carriage Return (ASCII 13dec)

3 Message dialogues

All device types that the protocol can handle support static and dynamic data. The protocol defines static data requests and dynamic data requests to read and write these two sets of data.

If probes are connected via a VPI board, there are two possible configurations.

Probes of different device type on one VPI board channel
In this case the "standard" requests can be used as it is enough to address the VPI board, the VPI channel and the device type to address an individual probe.

Probes of the same device type on one VPI channel
In this case the "serial number data field" must be additionally included in both, request and response of the message dialogue. Each device has a unique serial number which allows to differentiate probes of the same device type connected to a common VPI channel.
In the following a request / response message including the "serial number data field" is called an "Optional" request / response.

Probes/devices connected directly (without VPI)
If the probes/devices are directly connected (e.g. in case of TORRIX RS485) the multiplexer board address and channel (AC) must still be included in both, request and response of the message dialogue. In this case use multiplexer board address 1 and multiplexer channel 1, resulting in an AC value of 00H.

VISY-Input/VISY-Output
For VISY-Input and VISY-Output it is required to set the multiplexer board address. The board address can be set via DIP switches in the range 18 to 32. The channel address will not be regarded and should always be set to 1.

3.1 General message structure

Message structure: <G or F or X or Y> <AC> <D> [<data_field_1> … <data_field_n>] <:> <cs…cs> <cr>

The need for any data fields depends on the message type.

Each data field consists of an identifier (= ID) followed by a value: <<ID_1><Value_1>>.

The colon character <:> indicates the checksum to follow.

The carriage return character <cr> represents the end of the message.

3.2 Static data

There are message dialogues for reading and writing static data. These dialogues are described in the following sub-chapters.

3.2.1 Read static data

The device that receives a static data read request will send a static data read response that contains all static data fields (<ID><Value>) supported by the device.

Static data read request

Standard static data read request:

```
G AC D : cs cs cr
```

Example:

Board Address = 1, Channel = 2 ➔AC = 01H
Device Type = ‘a’

```
G 0 1 a : 2 A cr
```

```
47H, 30H, 31H, 61H, 3AH, 32H, 41H, 0DH
```

Optional static data read request: (like "Standard", but including the "serial number data field")

```
G AC D # SN : cs cs cr
```

Example:

Board Address = 1, Channel = 2 ➔AC = 01H
Device Type = ‘a’, Serial Number = 34594

```
G 0 1 a # 3 4 5 9 4 : 6 5 cr
```

```
47H, 30H, 31H, 61H, 23H, 33H, 34H, 35H, 39H, 34H, 3AH, 36H, 35H, 0DH
```

Static data read response

Standard static data read response:

```
G AC D <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

Optional static data read response: (like "Standard", but including the "serial number data field")

```
G AC D # SN <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

3.2.2 Write static data

The static data write request is used to write new values into one or more static data fields of the addressed device. If a received static data field is writable and the received value for this data field is within the defined valid range the device will overwrite the existing value with the new value. In the static data write response, the device will include all static data IDs that have been received. The values returned are the current values of the device.

Note that most devices will not respond to a static data write request and that most of the static data fields are read-only.

Response if "execution not possible"
Depending on the devices' UDP protocol implemention, the device will respond in a way saying that it has received the "Static data write request" but the write access to regarded variable / data field cannot be executed. In this case the device will return the regarded data field with the requested ID followed by a value of "minus zero" (-0).

Static data write request

Standard static data write request:

```
X AC D <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cr
```

Example:

Board Address = 18, Channel = 1 ➔ AC = 88H
Device Type = 'o'
write data field "*Hold Time After Communication Loss*": ID = 'h', value = 120 sec
write data field "*Option Flags*": ID= 'o', value = 0EH
(meaning: Maintenance Mode = 0, Output action after hold time = 1, Relay mode = 1, Relay delay = 1)

```
X 8 8 o h 1 2 0 o 0 E : 4 C cr
```

```
58H, 38H, 38H, 6FH, 68H, 31H, 32H, 30H, 6FH, 30H, 45H, 3AH, 34H, 43H, 0DH
```

Optional static data write request:

```
X AC D # SN <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cr
```

Example:

Board Address = 22, Channel = 1 ➔ AC = A8H
Device Type = 'o', Serial Number = 6985
write data field "*Hold Time After Communication Loss*": ID = 'h', value = 0 sec
write data field "*Option Flags*": ID = 'o', value = 04H
(meaning: Maintenance Mode = 0, Output action after hold time = 0, Relay mode = 1, Relay delay = 0)

```
X A 8 o # 6 9 8 5 h 0 o 0 4 : C 6 cr
```

```
58H, 41H, 38H, 6FH, 23H, 36H, 39H, 38H, 35H, 68H, 30H, 6FH, 30H, 34H, 3AH,
43H, 36H, 0DH
```

Static data write response

Standard static data write response:

```
X AC D <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

Optional static data write response:

```
X AC D # SN <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

3.3 Dynamic data

There are message dialogues for reading and writing dynamic data. These dialogues are described in the following sub-chapters.

3.3.1 Read dynamic data

The device that receives a dynamic data read request will send a dynamic data read response that contains all dynamic data fields (<ID><Value>) supported by the device.

Dynamic data read request

Standard dynamic data read request:

```
F AC D : cs cs cr
```

Example:

Board Address = 1, Channel = 3 ➔ AC = 02H
Device Type = 'b'

```
F 0 2 b : 6 2 cr
```

```
46H, 30H, 32H, 62H, 3AH, 36H, 32H, 0DH
```

Optional dynamic data read request:

```
F AC D # SN : cs cs cr
```

Example:

Board Address = 2, Channel = 6 ➔ AC = 0DH
Device Type = ‘b’, Serial Number = 44389

```
F 0 D b # 4 4 3 8 9 : 1 D cr
```

```
46H, 30H, 44H, 62H, 23H, 34H, 34H, 33H, 38H, 39H, 3AH, 31H, 44H, 0DH
```

Dynamic data read response

Standard dynamic data read response:

```
F AC D <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

Optional read dynamic data read response:

```
F AC D # SN <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

3.3.2 Write dynamic data

The dynamic data write request is used to write new values into one or more dynamic data fields of the addressed device. If a received dynamic data field is writable and the received value for this data field is within the defined valid range the device will overwrite the existing value with the new value. In the dynamic data write response, the device will include all dynamic data IDs that have been received. The values returned are the current values of the device.

Note that most devices will not respond to a dynamic data write request and that most of the dynamic data IDs are read-only.

Response if "execution not possible"
Depending on the devices' UDP protocol implemention, the device will respond in a way saying that it has received the "Dynamic data write request" but the write access to regarded variable / data field cannot be executed (e.g. the variable is unknown or currently not available). In this case the device will return the regarded data field with the requested ID followed by a value of "minus zero" (-0).

Dynamic data write request

Standard dynamic data write request:

```
Y AC D <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cr
```

Example:

Board Address = 30, Channel = 1 ➔ AC = E8H
Device Type = ‘o’
write data field "*Channel Data*": ID = 'c', value = 20H (meaning: activate Output 6 only)

```
Y E 8 o c 2 0 : A C cr
```

```
59H, 45H, 38H, 6FH, 63H, 32H, 30H, 3AH, 41H, 43H, 0DH
```

Optional dynamic data write request: (like "Standard", but including the "serial number data field")

```
Y AC D # SN <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cr
```

Example:

Board Address = 27, Channel = 1 ➔ AC = D0H
Device Type = ‘o’, Serial Number = 7993
write data field "*Channel Data*": ID = 'c', value = E1H (meaning: activate Outputs 1, 6, 7 and 8)

```
Y D 0 o # 7 9 9 3 c E 1 : B B cr
```

```
59H, 44H, 30H, 6FH, 23H, 37H, 39H, 39H, 33H, 63H, 45H, 31H, 3AH, 42H, 42H,
0DH
```

Dynamic data write response

Standard dynamic data write response:

```
Y AC D <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

Optional dynamic data write response: like "Standard", but including the "serial number data field"

```
Y AC D # SN <ID1> <VALUE1> … <IDn> <VALUEn> : cs cs cs cs cr
```

4 Device types

The different device types that the protocol can handle are identified by a unique device type ID. The following table shows the currently defined device types.

Device type IDs	
ID	**Description**
‘a’ 61H	**VISY-Stick and TORRIX…** Available since protocol version: 1.00
‘b’ 62H	**VISY-Stick/Reed Interstitial** Available since protocol version: 1.00
‘c’ 63H	**VISY-Stick/Reed Sump Manhole** Available since protocol version: 1.00
‘d’ 64H	**VISY-Stick/Reed Sump Dispenser** Available since protocol version: 1.00
‘e’ 65H	**VISY-Stick Density Only (reserved, not yet implemented)** Available since protocol version: 1.00
‘f’ 66H	**VISY-Stick Oil Separator (reserved, not yet implemented)** Available since protocol version: 1.01
‘g’ – ‘h’	Free for future expansions.
‘i’ 69H	**VISY-Input** Available since protocol version: 1.10
‘j’ – ‘k’	Free for future expansions.
‘l’ 6CH	**VIMS Product Line** Available since protocol version: 1.10
‘m’ 6DH	**VIMS Tank** Available since protocol version: 1.10
‘n’ 6EH	**VIMS Filling Line** Available since protocol version: 1.10
‘o’ 6FH	**VISY-Output** Available since protocol version: 1.10
‘p’ 70H	**Pressure Sensor** Available since protocol version: 1.04
‘q’	Free for future expansions.
‘r’ 72H	**Wireless Receiver - RFR (reserved, not yet implemented)** Available since protocol version: 1.10
‘s’ 73H	**VISY-Sludge** Available since protocol version: 1.08
‘t’ 74H	**VISY-Temp** Available since protocol version: 1.09
‘u’	Free for future expansions.
‘v’ 76H	**VPI (reserved, not yet implemented)** Available since protocol version: 1.10
‘w’ 77H	**Wireless Transmitter - RFT (reserved, not yet implemented)** Available since protocol version: 1.10

5 Data fields

Each data field consists of an ID directly followed by the value for this ID: <**ID**><**Value**>. Currently the characters '=', '#' and 'a' to 'w' are defined as data field IDs. In the future, it might be possible that this range will be expanded by special characters like "%".

Notes on interpreting data fields:

The sequence and number of the data fields (consisting of ID and value) in the response messages is not fixed.

Exception: Serial number data field. If the serial number data field is included, it follows directly the device type character (D) and its meaning is common for both, static and dynamic data responses.

Static and dynamic data responses use the same IDs but with different meaning.

If unknown IDs are present in a response message these IDs and the corresponding values should be ignored.

The field width of the individual values is not fixed. So, when interpreting a response message, the field width of a certain value results out of the number of characters between the corresponding ID and the next ID or ':' character that directly follows the value.

A value of -0 indicates that due to an error this value is currently not available (e.g. defective temperature sensor).

5.1 Static data fields

Static data fields are reported in the response to a static data read request. In general, the static data are read-only. Variables / data fields that are writable have a special note in the description.

The following table shows the currently defined static data fields.

Static data fields	
ID	**Data field Description**
'#' 23H	**Serial Number** This data field represents the serial number of the device. Resolution: 1 Valid range: 1 – 16777215 Example: A serial number of 431725 is transferred as 34H, 33H, 31H, 37H, 32H, 35H. Device types: all Available since protocol version: 1.00
'a' – 'c'	Free for future expansions.
'd' 64H	**Position Of Density Module** This data field represents the position of a density module mounted on the probe tube. Resolution: 1 mm Valid range: 0 – 65535 mm Example: A position of 250 mm is transferred as 32H, 35H, 30H. Device types: VISY-Stick and TORRIX... (device type ID = 'a'), VISY-Stick Density Only (device type ID = 'e') Available since protocol version: 1.00 **Notes:** This data field is only supported if the probe is equipped with at least one density module. If the probe is equipped with more than one density module, this data field is included more than once in the response message. In this case the position of the module with the highest mounting position (nearest to the head of the probe) is always transferred first and the position of the module with the lowest mounting position is transferred last. The measurement values of the density modules are transferred in the same sequence as their corresponding mounting positions.

Static data fields	
ID	**Data field Description**
‘e’ – ‘g’	Free for future expansions.
‘h’ 68H	**Hold Time After Communication Loss** The hold time after communication loss defines if and when the outputs of a VISY-Output react after a communication loss occurred. Resolution: 1 sec Valid range: 0 – 240 sec Example: A hold time of 120 sec is transferred as 31H, 32H, 30H. Device types: VISY-Output (device type ID = ‘o’) Available since protocol version: 1.10 **Notes:** For device type VISY-Output this data field is writable. If the hold time is set to 0 then the outputs will keep the last status that they had before the communication loss occurred. For further information about the hold time please also see the Technical Documentation of the output device.
‘i’ 69H	**Alarm Pressure** This data field represents the alarm pressure/vacuum. If the measured pressure in the interstitial space falls below this value an alarm is signalized. For device types VIMS Product Line (device type ID = ‘l’), VIMS Tank (device type ID = ‘m’) and VIMS Filling Line (device type ID = ‘n’): Resolution: 1 mbar (1 hPa) Valid range: -1000 mbar … +1000 mbar Example: An alarm pressure of -500 mbar is transferred as 2DH, 35H, 30H, 30H. Available since protocol version: 1.10 **Notes:** 0 Pa and all positive pressures are transferred without leading ‘+’ character.
‘j’ – ‘k’	Free for future expansions.

Static data fields	
ID	**Data field Description**
‘l’ 6CH	**Probe Length** This data field represents the length of the probe. Resolution: 1 mm Valid range: 0 – 65535 mm Example: A probe length of 15000 mm is transferred as 31H, 35H, 30H, 30H, 30H. Device types: all Available since protocol version: 1.00
‘m’ – ‘n’	Free for future expansions.

Static data fields	
ID	**Data field Description**
'o' 6FH	**Option Flags** The option flags can be used to configure the outputs / relays of VISY-Output to adopt their functions to the requirements of the application. This ID is bit based (bit 0 – bit 7). Bit 0 – Maintenance mode = 0 → Maintenance mode off = 1 → Maintenance mode on Bit 1 – Output action after hold time = 0 → Output will be deactivated = 1 → Output will be activated Bit 2 – Relay mode = 0 → Standard mode = 1 → Failsafe mode Bit 3 – Relay delay = 0 → Relay delay off = 1 → Relay delay on Bit 4...7 – free for future expansions Format: XX (00H...FFH) Example: When the bits are set like this: bit 0 = 0, bit 1 = 1, bit 2 = 1, bit 3 = 1 and bits 4...7 = 0 (unused) then 30H, 45H will be transferred. Device types: VISY-Output (device type ID = 'o') Available since protocol version: 1.10 **Notes:** For device type VISY-Output this data field is writable. For further information about the option flags please also see the Technical Documentation of the output device.

Static data fields	
ID	**Data field Description**
‘p’ 70H	**Protocol Version** This data field represents the version of this protocol that is implemented in the device. Format: XX XX (00H…FFH . 00H…FFH) Example: A protocol version of 01.07 (01H . 07H) is transferred as 30H, 31H, 30H, 37H. Device types: all Available since protocol version: 1.00
‘q’ – ‘r’	Free for future expansions.
‘s’ 73H	**Maximum Distance** This data field represents the maximum distance that the sensor VISY-Sludge can measure. Resolution: 1 mm Valid range: 0 – 65535 mm Example: A maximum distance of 1000 mm is transferred as 31H, 30H, 30H, 30H. Device types: VISY-Sludge (device type ID = ‘s’), Available since protocol version: 1.08

Static data fields	
ID	**Data field Description**
‘t’ 74H	**Position Of Temperature Sensor** This data field represents the mounting position of a temperature sensor within a probe. Resolution: 1 mm Valid range: 0 – 65535 mm Example: A position of 2850 mm is transferred as 32H, 38H, 35H, 30H. Device types: VISY-Stick and TORRIX... (device type ID = ‘a‘), VISY-Stick Density Only (device type ID = ‘e‘) VISY-Temp (device type ID = ‘t‘) Available since protocol version: 1.00 **Note:** This data field is only supported if the probe is equipped with at least one discrete temperature sensor. This data field can be included more than once in the response message depending on the number of temperature sensors. In this case the position of the temperature sensor with the lowest mounting position (nearest to the foot point of the probe) is always transferred first and the position of the temperature sensor with the highest mounting position is transferred last. The measurement values of the temperature sensors are transferred in the same sequence as their corresponding mounting positions.

Static data fields	
ID	**Data field Description**
‘u’ 75H	**Device Sub-Type** The device sub-type is a supplement to the device type that describes the specific characteristic of the device. The following sub-types are currently defined: For device type VISY-Stick (device type ID = ‘a’) • 1 – Basic • 2 – Standard • 3 – Advanced • 4 – Flex available since protocol version: 1.00 For device types VISY-Stick/ Reed Interstitial (device type ID = ‘b’), VISY-Stick/ Reed Sump Manhole (device type ID = ‘c’) and VISY-Stick/ Reed Sump Dispenser (device type ID = ‘d’) • 1 – Stick based (magnetostrictive) • 2 – Reed based (read switches) available since protocol version: 1.00 For device type VISY-Stick Density Only (device type ID = ‘e’) • 1 – Basic • 2 – Standard • 3 – Advanced available since protocol version: 1.00 For device type Pressure Sensor (device type ID = ‘p’) • 1 – VPS-V • 2 – VPS-L • 3 – VPS-T VPS-V and VPS-L available since protocol version: 1.05 VPS-T available since protocol version: 1.07 For device types VISY-Input (device type ID = 'i') and VISY-Output (device type ID = 'o') • 1…8 – number of input/output channels available since protocol version: 1.10

Static data fields	
ID	**Data field Description**
‘v’ 76H	**Firmware Version** This data field represents the firmware version of the device. Format: XX XX XX XX (00H…FFH . 00H…FFH . 00H…FFH . 00H…FFH) Example: A firmware version of 17.5.1.255 (11H . 05H . 01H . FFH) is transferred as 31H, 31H, 30H, 35H, 30H, 31H, 46H, 46H. Device types: all Available since protocol version: 1.00
‘w’	Free for future expansions.

5.2 Dynamic data fields

Dynamic data fields are reported in the response to a dynamic data read request. In general, the dynamic data are read-only. Writable data fields have a special note in the description.

The following table shows the currently defined dynamic data fields.

Dynamic data fields	
ID	**Description**
‘=’ 3DH	**Status Of Device** This data field represents the status of the device. The following values are currently defined: • 0 – OK, the device is working well • 1 – Error, the device has detected an internal error and is not working well Device types: all Available since protocol version: 1.02 **Notes:** If the status of the device is not reported to be ‘0’ this should be handled as an error. If the status of the device is reported to be ‘1’ then no further data IDs will be included in the dynamic data response.

Dynamic data fields	
ID	**Description**
‘a’ 61H	**Alarm** This data field represents an alarm that the device wants to report. The following alarms are currently defined: For device types VISY-Stick/ Reed Interstitial (device type ID = ‘b’) VISY-Stick/ Reed Sump Manhole (device type ID = ‘c’) VISY-Stick/ Reed Sump Dispenser (device type ID = ‘d’) • 1 – Tamper alarm (only VISY-Stick based probes) • 2 – Fuel alarm (only VISY-Stick based probes) • 3 – High level alarm (only VISY-Reed based probes) • 4 – Low level alarm (only VISY-Reed based probe of probe type ‘b’) available since protocol version: 1.00 For device types VIMS Product Line (device type ID = ‘l’) VIMS Tank (device type ID = ‘m’) VIMS Filling Line (device type ID = ‘n’) • 1 – Alarm detected (this flag is activated when any kind of alarm situation as detailed in the other alarm flags has been detected) • 2 – Alarm pressure reached (the pressure in the interstitial space has reached the alarm pressure) • 3 – Product detected (product has been detected in the interstitial space) • 4 – Liquid detected (any liquid has been detected in the interstitial space; this flag is also activated when product has been detected) • 5 – No vacuum build-up (it is not possible to generate sufficient vacuum in the interstitial space; the vacuum source is defect/not running or the solenoid valve does not open) • 6 – Overpressure detected (an overpressure of more than +500 mbar has been detected in the interstitial space) available since protocol version: 1.10 **Notes:** This data field is only included in the response message if there is at least one active alarm. In case of multiple active alarms, each active alarm will be represented by its own alarm data field in the message.

Dynamic data fields	
ID	**Description**
‘b’ 62H	**Battery Status** This data field represents the status of the battery located in a device (e.g. RFT). Resolution: 1 Format: XX (00H...64H) Valid range: 0 – 100 • 0 = unknown • 1...100 = very low ... very high Example: A battery status of 32 (20H) is transferred as 32H, 30H. Device types: devices connected wireless Available since protocol version: 1.10 **Note:** This data field is only supported if the device is connected wireless.
‘c’ 63H	**Channel Data** This data field represents the current status of up to eight inputs or outputs. Format: XX (00H...FFH) Information is bit-based: bit 0 ≙ Input 1 / Output 1 bit 1 ≙ Input 2 / Output 2 ::: ::: ::: bit 7 ≙ Input 8 / Output 8 If the device supports less than 8 inputs or outputs, only the lower-order bits are significant. Example: VISY-Output 4: bits 0...3 are significant, bits 4...7 are constantly 0. Valid range: • bit value = 0: inactive • bit value = 1: active Example: A VISY-Input 8 device with Input 6 active, only, will respond a value of 20H transferred as 32H, 30H. Device types: VISY-Input (device type ID = 'i') and VISY-Output (device type ID = 'o') Available since protocol version: 1.10 **Note:** For device type VISY-Output (device type ID = 'o') this data field is writable.

Dynamic data fields	
ID	**Description**
‘d’ 64H	**Density** This data field represents the measured density value. Resolution: 0.1 g/l Valid range: 0 – 9999.9 g/l Example: A density of 769.8 g/l is transferred as 37H, 36H, 39H, 38H. Device types: VISY-Stick and TORRIX... (device type ID = 'a') VISY-Stick Density Only (device type ID = 'e') Available since protocol version: 1.00 **Notes:** This data field is only supported if the probe is equipped with at least one density module. If the probe provides multiple density values, each density value will be represented by its own data field in the message.

Dynamic data fields	
ID	**Description**
‘e’ 65H	**Event** This data field represents an event that the device wants to report. The following events are currently defined: For device type VISY-Stick (device type ID = 'a') • 1 – Start-Up (reduced measuring accuracy) • 2 – Filling detected available since protocol version: 1.00 For device type TORRIX… (device type ID = 'a') • 1 – Start-Up (reduced measuring accuracy) available since protocol version: 1.00 For device type TORRIX…VT (device type ID = 'a') • 1 – Start-Up (reduced measuring accuracy) • 3 – Raw product level data (strong waves on the media surface, wave filter is not active, reduced measurement accuracy) available since protocol version: 1.10 For device type VISY-Sludge (device type ID = 's') • 1 – Start-Up or no current distance value available available since protocol version: 1.08 For device types VIMS Product Line (device type ID = ‘l’), VIMS Tank (device type ID = ‘m’) and VIMS Filling Line (device type ID = ‘n’) • 1 – Solenoid valve open (the solenoid valve has been opened to maintain the vacuum in the interstitial space) • 2 – Vacuum Source active (it has been detected that the vacuum source is active) • 3 – Requesting Vacuum (this is a request to a higher system to start the vacuum source so that the vacuum in the interstitial space can be maintained) available since protocol version: 1.10 **Notes:** This data field is only included in the response message if there is at least one active event. In case of multiple active events, each active event will be represented by its own data field in the message.

Dynamic data fields	
ID	**Description**
‘f’ 66H	**Field Strength** This data field represents the reception field strength that the wireless receiver has measured. Resolution: 1 Format: XX (00H…64H) Valid range: 0 – 100 0 = unknown 1…100 = very low … very high Example: A reception field strength of 34 (22H) is transferred as 32H, 32H. Device types: devices connected wireless Available since protocol version: 1.10 **Note:** This data ID is only supported if the device is connected wireless.
‘g’ – ‘h’	Free for future expansions.
‘i’ 69H	**Pressure** This data field represents the measured pressure/vacuum. The following pressure/vacuum measurements are currently defined: For device type Pressure Sensor (device type ID = ‘p’) Resolution: depends on the Device Sub-Type of the Pressure Sensor • VPS-V and VPS-T: 1 µbar (0.1 Pa) • VPS-L: 1 mbar (100 Pa) Valid range: not limited, depends on the used sensor Example for VPS-V and VPS-T: A pressure of 14.763 mbar (1476.3 Pa) is transferred as 31H, 34H, 37H, 36H, 33H. Example for VPS-L: A pressure of 2.861 bar (2861 hPa) is transferred as 32H, 38H, 36H, 31H. Available since protocol version: 1.04 (corrected in 1.07) For device types VIMS Product Line (device type ID = ‘l’), VIMS Tank (device type ID = ‘m’) and VIMS Filling Line (device type ID = ‘n’) Resolution: 0.1 mbar (0.1 hPa) Valid range: -1000.0 mbar … +1000.0 mbar Example: A pressure of -305.7 mbar is transferred as 2DH, 33H, 30H, 35H, 37H. Available since protocol version: 1.10 **Notes:** 0 Pa and all positive pressures are transferred without leading ‘+’ character.
‘j’ – ‘o’	Free for future expansions.

Dynamic data fields	
ID	**Description**
‘p’ 70H	**Product Level** This data field represents the level of the product. Resolution: 1 µm Valid range: 0 – 65535 mm Example: A Product Level of 1367.5 mm is transferred as 31H, 33H, 36H, 37H, 35H, 30H, 30H. Device types: VISY-Stick and TORRIX... (device type ID = 'a') Available since protocol version: 1.00 **Notes:** For probes of device type VISY-Stick and TORRIX... (device type ID = 'a') to get the real surface level of the product a correction must be done as the real level depends on the density of the product. For device type VISY-Stick Oil Separator (device type ID = 'f') this Data ID represents the level at the interface between oil and water.
‘q’	Free for future expansions.
‘r’ 72H	**Age Of Data** This data field indicates how old the data is. Resolution: 1 sec Format: XXXXX (0x00000...0xFFFFF) Valid range: 1 – 1048575 sec (about 12 days) Example: If the age of data is 384 (0x180) sec then 31H, 38H, 30H will be transferred. Available since protocol version: 1.10 **Note:** This Data ID is usually supported if the device is connected wireless.

Dynamic data fields	
ID	**Description**
‘s’ 73H	**Distance** This data field represents the distance measured from the bottom of the sensor VISY-Sludge to the reflection surface. Resolution: 0.1 mm Valid range: 0 – 65535 mm Example: A distance of 243.7 mm is transferred as 32H, 34H, 33H, 37H. Device types: VISY-Sludge (device type ID = 's') Available since protocol version: 1.08
‘t’ 74H	**Temperature** This data field represents the temperature. Resolution: 0.001 °C Valid range: -99.999 – 999.999 °C Example: A temperature of -14.2 °C is transferred as 2DH, 31H, 34H, 32H, 30H, 30H Device types: VISY-Stick and TORRIX... (device type ID = 'a') VISY-Stick Density Only (device type ID = 'e') Pressure Sensor (device type ID = ‘p’) VISY-Sludge (device type ID = ‘s’) VISY-Temp (device type ID = ‘t’) Available since protocol version: 1.00 **Notes:** 0.000 °C and all positive temperatures are transferred without leading ‘+’ character. This data ID can be included more than once in the response message if the device supports more than one temperature sensor.
‘u’	Free for future expansions.

Dynamic data fields	
ID	**Description**
‘v’ 76H	**Tightness Indicator** This data field value informs about how gas tight the pneumatic installation of the vacuum interstice monitoring system is. Resolution: 1 Valid range: 0 – 10 (with 0 = totally tight to 10 = totally untight) Example: A tightness indicator of 4 is transferred as 34H. Device types: VIMS Product Line (device type ID = ‘l’) VIMS Tank (device type ID = ‘m’) VIMS Filling Line (device type ID = ‘n’) Available since protocol version: 1.10
‘w’ 77H	**Water Level** This data field represents the level of water. Resolution: 0.1 mm Valid range: 0 – 65535 mm Example: A water level of 51.0 mm is transferred as 35H, 31H, 30H. Device types: VISY-Stick and TORRIX... (device type ID = 'a') VISY-Stick/ Reed Interstitial (device type ID = ‘b’) VISY-Stick/ Reed Sump Manhole (device type ID = ‘c’) VISY-Stick/ Reed Sump Dispenser (device type ID = ‘d’) Available since protocol version: 1.00 **Notes:** This data field is only included in the response message if the device is equipped with a water float. For probes of device type VISY-Stick and TORRIX... (device type ID = 'a') to get the real surface level of the water a correction must be done as the real level depends on the density of the product. For VISY-Stick based probes of device type IDs ‘b’, ‘c’ and ‘d’ this data field represents a general liquid level and not stringently a water level. For VISY-Reed based probes of device type IDs ‘b’, ‘c’ and ‘d’ this Data field is not included in the response message.

6 Attachment A – CRC calculation

The following routines demonstrate how the CRC calculation could be done.

C routine

```
unsigned int CalcCRC16(unsigned char* data, unsigned int len) {

unsigned int i, j, x16, crc;
unsigned char input;

  crc = 0;                                    // crc start value
  for (i = 0; i < len; i++) {
    input = *(data++);                        // get next data
    for (j = 0; j < 8; j++) {
      if ((crc & 0x0001) ^ (input & 0x01))  // determine the x16 value
        x16 = 0x8408;                         // generator polynominal
      else
        x16 = 0x0000;
      crc = crc >> 1;                         // shift crc
      crc = crc ^ x16;                        // xor in the x16 value
      input = input >> 1;                     // shift input for next iteration
    }
  }
  return(crc);
}
```

Pascal function

```
(With “array of char”)

function TForm1.CalcCRC16(data: array of char; count: word): word;
var
  i : integer;
  crclow, crchigh : byte;
  j, k : word;
begin
  crclow := 0;
  crchigh := 0;
  for i := 0 to (count - 1) do begin
    j := crclow xor Word(data[i]);
    k := ((j shr 4) xor j) and $000f;
    j := (j and $000f) OR (k shl 4);
    crclow := Byte((crchigh xor (j shl 3)) xor k);
    crchigh := Byte(j xor (k shr 1));
  end; // for
  result := Word((crchigh * 256) + crclow);
end;
```

```
(With “string”)

function TForm1.CalcCRC16(data: string; count: word): word;
var
  i : integer;
  crclow, crchigh : byte;
  j, k : word;
begin
  crclow := 0;
  crchigh := 0;
  for i := 1 to count do begin
    j := crclow xor Word(data[i]);
    k := ((j shr 4) xor j) and $000f;
    j := (j and $000f) OR (k shl 4);
    crclow := Byte((crchigh xor (j shl 3)) xor k);
    crchigh := Byte(j xor (k shr 1));
  end; // for
  result := Word((crchigh * 256) + crclow);
end;
```

Visual Basic function

```
Public Function getChkCrc16(ByVal strText As String) As UShort
Dim i, j, ii, x16, crc As UShort
Dim Input As Byte

'= for the XOR we need an array of byte. So let's convert it here.
Dim chrData() As Char = strText.ToCharArray
Dim bytData() As Byte
ReDim bytData(chrData.Length - 1)

Try
  For ii = 0 To chrData.Length - 1
    bytData(ii) = Convert.ToByte(chrData(ii))
  Next

  ii = 0                              ' Index to bytData
  crc = 0                             ' crc start value (seed)
  For i = 0 To bytData.Length - 1
    Input = bytData(i)                ' get next data byte
    For j = 0 To 7
      If ((crc And &H1) Xor (Input And &H1)) Then 'determine the x16 value
        x16 = &H8408                  ' generator polynom
      Else
        x16 = &H0
      End If
      crc = crc >> 1                  ' shift crc
      crc = crc Xor x16               ' XOR with the generator polynome or zero
      Input = Input >> 1
    Next
  Next

Catch ex As Exception
  crc = &H1234
  MessageBox.Show("Error: " & ex.Message)
End Try
 Return crc
End Function
```

7 Attachment B – Supported data fields

The following tables show which device supports which static and dynamic data fields.

Static data fields

Supported Static Data IDs sorted by Device Type

Device Type	Device Sub-Type 'u'	Serial Number '#'	Firmware Version 'v'	Protocol Version 'p'	Probe Length 'l'	Maximum Distance 's'	Position Of Temperature Sensor 't'	Position Of Density Module 'd'	Hold Time 'h'	Option Flags 'o'	Alarm Pressure 'i'
VISY-Stick and TORRIX.. ('a')	yes	yes	yes	yes	yes	no	yes	no/yes *1	no	no	no
VISY-Stick Interstitial ('b')	yes	yes	yes	yes	yes	no	no	no	no	no	no
VISY-Reed Interstitial ('b')	yes	yes	yes	yes	yes	no	no	no	no	no	no
VISY-Stick Sump Manhole ('c')	yes	yes	yes	yes	yes	no	no	no	no	no	no
VISY-Reed Sump Manhole ('c')	yes	yes	yes	yes	yes	no	no	no	no	no	no
VISY-Stick Sump Dispenser ('d')	yes	yes	yes	yes	yes	no	no	no	no	no	no
VISY-Reed Sump Dispenser ('d')	yes	yes	yes	yes	yes	no	no	no	no	no	no
VISY-Stick Density Only ('e')	yes	yes	yes	yes	yes	no	yes	yes	no	no	no
VISY-Input ('i')	yes	yes	yes	yes	no	no	no	no	no	no	no
VIMS Product Line ('l')	no	yes	yes	yes	no	no	no	no	no	no	yes
VIMS Tank ('m')	no	yes	yes	yes	no	no	no	no	no	no	yes
VIMS Filling Line ('n')	no	yes	yes	yes	no	no	no	no	no	no	yes
VISY-Output ('o')	yes	yes	yes	yes	no	no	no	no	yes *2	yes *2	no
Pressure Sensor ('p')	yes	yes	yes	yes	no	no	no	no	no	no	no
VISY-Sludge ('s')	no	yes	yes	yes	no	yes	no	no	no	no	no
VISY-Temp ('t')	no	yes	yes	yes	no	no	yes	no	no	no	no

NOTE: In general all Static Data Ids are read-only.

yes	always supported
no	never supported
no/yes *1	only supported if at least one density module is mounted
yes *2	this Data ID is writable

Dynamic data fields

Supported Dynamic Data Ids sorted by Device Type

Device Type	Status Of Device '='	Product Level 'p'	Water Level 'w'	Distance 's'	Temperature 't'	Density 'd'	Pressure 'i'	Alarm 'a'	Event 'e'	Channel Data 'c'	Tightness Indicator 'v'
VISY-Stick and TORRIX.. ('a')	yes	yes	yes/no *2	no	yes	no/yes *3	no	no	yes	no	no
VISY-Stick Interstitial ('b')	yes	no	yes *1	no	no	no	no	yes	no	no	no
VISY-Reed Interstitial ('b')	yes	no	no	no	no	no	no	yes	no	no	no
VISY-Stick Sump Manhole ('c')	yes	no	yes *1	no	no	no	no	yes	no	no	no
VISY-Reed Sump Manhole ('c')	yes	no	no	no	no	no	no	yes	no	no	no
VISY-Stick Sump Dispenser ('d')	yes	no	yes *1	no	no	no	no	yes	no	no	no
VISY-Reed Sump Dispenser ('d')	yes	no	no	no	no	no	no	yes	no	no	no
VISY-Stick Density Only ('e')	yes	no	no	no	yes	yes	no	no	no	no	no
VISY-Input ('i')	yes	no	no	no	no	no	no	no	no	yes	no
VIMS Product Line ('l')	yes	no	no	no	no	no	yes	yes	yes	no	yes
VIMS Tank ('m')	yes	no	no	no	no	no	yes	yes	yes	no	yes
VIMS Filling Line ('n')	yes	no	no	no	no	no	yes	yes	yes	no	yes
VISY-Output ('o')	yes	no	no	no	no	no	no	no	no	yes *4	no
Pressure Sensor ('p')	yes	no	no	no	yes	no	yes	no	no	no	no
VISY-Sludge ('s')	yes	no	no	yes	yes	no	no	no	yes	no	no
VISY-Temp ('t')	yes	no	no	no	yes	no	no	no	no	no	no

NOTE: In general all Dynamic Data Ids are read-only.

yes	always supported
no	never supported
yes *1	in this case the Data ID represents a general liquid level
yes/no *2	not supported if no water float is mounted
no/yes *3	only supported if at least one density module is mounted
yes *4	this Data ID is writable

The following Dynamic Data IDs are additionally supported by all devices that are connected wireless:

- 'b' - Battery Status
- 'f' - Field Strength
- 'r' - Age Of Data

8 Attachment C – Revision history

The following table shows the revision history of this document.

Protocol Version	Modification	Date	Name
1.00	First release	2010-11-30	KR
1.01	Chapter 4, added new Device Type for VISY-Stick Oil Separator. Chapter 6.2, Device Type ‘f’ added to ID ‘p’ (Product Level). Chapter 6.2, Device Types ‘b’, ‘c’, ‘d’ and ‘f’ added to ID ‘w’ (Water Level).	2011-05-13	KR
1.02	Chapter 5 (Status Of Device), deleted. Chapter 3, format of request and response messages changed, checksum values added in the examples. Chapter 5.1, ID ‘p’ (Protocol Version), format changed. Chapter 5.2, ID ‘=’ (Status Of Device), now defined as part of the Dynamic Data IDs. Chapter 5.2, ID ‘t’ (Temperature), deleted Device Types ‘b’, ‘c’ and ‘d’. Chapter 5.2, ID ‘w’ (Water Level), added notes for Device Types ‘b’, ‘c’ and ‘d’. Chapter 6, added Pascal and Visual Basic examples for CRC calculation.	2011-06-01	UN, KR, WS, CM
1.03	Chapter 5.1, ID ‘u’ (Device Sub-Type), added Sub-Type 4 (Flex) for Device Type ‘a’.	2011-07-01	KR, CM
1.04	Chapter 4, added new Device Type ‘p’ for Pressure Sensor. Chapter 5.1, ID ‘d’ (Position Of Density Module), corrected the note for the case that more than one Density Module is configured. Chapter 5.2, added Dynamic Data ID ‘i’ (Pressure) for Device Type ‘p’. Chapter 5.2, ID ‘p’ (Product Level), added note for Device Type ‘a’. Chapter 5.2, ID ‘w’ (Water Level), added note for Device Type ‘a’.	2011-11-30	KR, CM
1.05	Attachment A, added Delphi routine with string. Chapter 5.1, ID ‘u’ (Device Sub-Type), added Sub-Type 1 (VPS-V) and Sub-Type 2 (VPS-L) for Device Type ‘p’. Chapter 5.2, ID ‘t’ (Temperature), added Device Type ‘p’.	2012-08-28	KR, MO
1.06	Chapter 2, added missing information and re-arranged chapter. Chapter 3, inserted new Chapter 3.1 (General message structure) Chapter 3.2, devided into sub-chapters 3.2.1 (Read Static Data) and 3.2.2 (Write Static Data). Chapter 3.3, devided into sub-chapters 3.3.1 (Read Dynamic Data) and 3.3.2 (Write Dynamic Data). Chapter 4, added new Device Type ‘i’ for VISY-Input. Chapter 4, added new Device Type ‘o’ for VISY-Output. Chapter 5.1, added Static Data ID ‘h’ (Hold Time After Communication Loss) for Device Type ‘o’. Chapter 5.1, added Static Data ID ‘o’ (Option Flags) for Device Type ‘o’. Chapter 5.2, removerd Status ‘?’ for Dynamic Data ID ‘=’ (Status Of Device). Chapter 5.2, added Dynamic Data ID ‘c’ (Channel State) for Device Types ‘i’ and ‘o’. Chapter 7, added Attachment B – Supported data fields.	2013-01-22	KR, WS, CM, UN
1.07	Chapter 5.1, Static Data ID ‘u’ (Device Sub-Type), added Sub-Type 3 (VPS-T) for Device Type ‘p’. Chapter 5.2, Dynamic Data ID ‘i’ (Pressure), corrected resolution and example.	2015-09-01	KR
1.08	Chapter 4, added new Device Type ‘s’ for VISY-Sludge. Chapter 5.1, added Static Data ID ‘s’ (Maximum Distance) for Device Type ‘s’. Chapter 5.2, added Dynamic Data ID ‘s’ (Distance) for Device Type ‘s’.	2016-09-16	KR

Protocol Version	Modification	Date	Name
	Chapter 5.2, ID ‘t’ (Temperature), added Device Type ‘s’. Chapter 5.2, ID ‘e’ (Event), added an event for Device Type ‘s’. Chapter 7, Attachment B, added data fields for VISY-Sludge.		
1.09	Chapter 3, added text regarding probes that are directly connected (without a multiplexer). Chapter 4, added TORRIX … to Device Type ‘a’. Chapter 4, added new Device Type ‘t’ for VISY-Temp Chapter 5.1, ID ‘t’ (Position Of Temperature Sensor), added Device Type ‘t’. Chapter 5.2, ID ‘p’ (Product Level), changed note for Device Type ‘f’. Chapter 5.2, ID ‘t’ (Temperature), added Device Type ‘t’. Chapter 7, Attachment B, added data fields for VISY-Temp.	2018-02-20	KR
1.10	Chapter 3, added general information about the addressing of Device Types ‘i’, and ‘o’ and added new examples. Chapter 4, added new Device Types VIMS Product Line (device type ID = ‘l’), VIMS Tank (device type ID = ‘m’) and VIMS Filling Line (device type ID = ‘n’). Chapter 4, added new Device Type ‘r’ for Wireless Receiver and ‘w’ for Wireless Transmitter. Chapter 5.1, added Static Data ID ‘i’ (Alarm Pressure) for Device Types ‘l’, ‘m’ and ‘n’. Chapter 5.1, Static Data ID ‘u’ (Sub-Type), added sub-types for Device Type ‘i’ and ‘o’. Chapter 5.2, Dynamic Data ID ‘a’ (Alarm), added alarms for Device Types ‘l’, ‘m’ and ‘n’. Chapter 5.2, Dynamic Data ID ‘c’ (Channel Data), changed the format of the data value, now all available channels are included in the data value. Chapter 5.2, Dynamic Data ID ‘e’ (Event), added events for TORRIX…VT. Chapter 5.2, Dynamic Data ID ‘e’ (Event), added events for Device Types ‘l’, ‘m’ and ‘n’. Chapter 5.2, Dynamic Data ID ‘i’ (Pressure), added pressure for Device Types ‘l’, ‘m’ and ‘n’. Chapter 5.2, added Dynamic Data ID ‘v’ (Tightness Indicator) for Device Types ‘l’, ‘m’ and ‘n’. Chapter 5.2, changed the Dynamic Data ID for Age Of Data from ‘o’ to ‘r’. Chapter 7, Attachment B, added data fields for Device Types ‘i’, and ‘o’.	2025-09-09	KR, LJ, WS

Blank page

QR code to the website
Technical Documentation

FAFNIR GmbH
Schnackenburgallee 149 c
22525 Hamburg, Germany
Tel.: +49 / 40 / 39 82 07–0
E-mail: info@fafnir.com
Web: www.fafnir.com