Nordic Automation Systems

LoRaWAN®
PULSER BK-G
CM3061

LoRaWAN® Pulser BK-G Elster diaphragm gas meters BK-G1,6 to BK-G100 that are equipped with the index Z3/Z6 and transmits it wirelessly to the LoRaWAN® network.

LoRaWAN® Pulser BK-G is meant to be connected to the existing gas meter's pulse output. It is powered by a long-life battery.

FW >= 1.2.1

OVERVIEW

Efficient

LoRaWAN® Pulser BK-G has a bidirectional, battery powered, long-range transceiver with low power consumption.

Intelligent

Real-time usage data is gathered wirelessly and processed automatically. Data is accessible from your LoRaWAN® provider.

APPLICATIONS

Pulse metering

Frequent reporting provides a detailed usage overview.

Tamper detection

LoRaWAN® Pulser BK-G can detetct magnetic tampering.

FEATURES

- Long range wireless data transmission
- Pulse counting
- Pre-installed long-life battery
- Configurable reporting interval
- Maintenance free - install & forget
- Easy installation
- Average life 8 years*
- Secure communication

* Lifetime depends from the device location and reporting interval.

SPECIFICATIONS

Length:	93.5 mm
Height:	48.5 mm
Width:	25 mm
Weight:	80 g
Operating temperature:	-20°C ... +65°C
Body material:	ABS
IP Rating:	IP68
ATEX Approved:	Ex II 3 G Ex ic IIA T5 Gc - EESF 19 ATEX 064X
Antenna connector:	SMA

COMMUNICATION SPECIFICATIONS

LoRAWAN®	Device class:	A
	Activation:	OTAA
	LoRAWAN® version:	1.0.3a
	Rx Sensitivity:	-140 dBm
	Tx power:	up to +20 dBm

wM-Bus	Mode:	C1 / T1
	OMS version:	4.1.2

INDEX

fPort 24 Status Message

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9..12
header	*general*	*active_alerts*	*battery*		*board*		*downlink*		*counter_instant*
			percentage	*voltage*	*temperature*	*extremes*	*rssi*	*snr*	
0x01	See fig. 1	See fig. 2	type: uint8 unit: % value: /2.54 0xFF - ***not_available***	type: uint8 unit: V value: mapped 3_6V	type: int8 unit: °C	See fig. 3	type: uint8 unit: dBm value: *-1	type: int8 unit: Bm	type: uint32 unit: l 0xFFFF FFFF - ***not_available***

1.

Bit #	Parameter	Value
0	RFU	
1	RFU	
2	RFU	
3	RFU	
4	RFU	
5	*packet_reason_app*	0 - *false* 1 - *true*
6	*packet_reason_magnet*	
7	*packet_reason_alert*	

2.

Bit #	Parameter
0	RFU
1	RFU
2	RFU
3	*tamper*
4	RFU
5	RFU
6	RFU
7	RFU

3.

Bit #	Parameter	Value
0	*min_offset*	type: uint4 unit: °C value: *-2
1		
2		
3		
4	*max_offset*	type: uint4 unit: °C value: *2
5		
6		
7		

Status message can be triggered with 1 second magnet switch (same place as activating). The time between two consecutive packets is dependent of the duty cycle. If the message is triggered too often, it will go to lockdown for 1 hour. It will still be working as meant to, but will not send any data.

Message sample

Message in base64

```
AYAI/5ISAFIHoAAAAA==
```

Message decoded to HEX

```
01|80|08|FF|92|12|00|52|07|A0000000
```

header HEX message

```
0x01
```

general HEX message

```
0x80
```

general HEX message converted to binary

```
0B10000000
```

Binary converted to statuses (LSB)

```
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : packet_reason_app - false
0 : packet_reason_magnet - false
1 : packet_reason_alert - true
```

active_alerts HEX message

```
0x08
```

active_alerts HEX message converted to binary

```
0B00001000
```

Binary converted to statuses (LSB)

```
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
1 : tamper - true
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

battery

percentage HEX message

```
0xFF
```

HEX message converted to value

```
not_available
```

voltage HEX message

```
0x92
```

HEX message converted to decimal

```
146
```

Decimal value mapped to voltage

```
3.246 (V)
```

board

temperature HEX message

```
0x12
```

HEX message converted to signed decimal

```
18 (°C)
```

extremes HEX message

```
0x00
```

extremes HEX message converted to binary

```
0B00000000
```

Binary converted to extremes (LSB)

```
0 : min_offset [0B0000]
0 :
0 :
0 :
0 : max_offset [0B0000]
0 :
0 :
0 :
```

min_offset binary value converted to HEX

```
0x00
```

HEX message converted to decimal

```
0
```

Decimal offset value multiplied by -2

```
0 (°C)
```

max_offset binary value converted to HEX

```
0x00
```

HEX message converted to decimal

```
0
```

Decimal offset value multiplied by 2

```
0 (°C)
```

downlink

rssi HEX message

```
0x52
```

HEX message converted to decimal

```
82
```

Decimal value multiplied by -1

```
-82 (dBm)
```

snr HEX message

```
0x07
```

HEX message converted to signed decimal

```
7 (dB)
```

counter_instant HEX message

```
0xA0000000
```

HEX message flip for MSB

```
0x000000A0
```

HEX message converted to decimal

```
160 (Liters)
```

fPort 25 Usage Message

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
header	*general*	*counter_instant*			
0x01	See fig. 1	type: uint32 unit: l 0xFFFFFFFF - ***not_available***			

1.

Bit #	Parameter	Value
0	RFU	
1	RFU	
2	*usage_detected*	0 - false 1 - true
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Message sample

Message in base64

```
AQRoAQAA
```

Message decoded to HEX

```
01|04|68010000
```

header HEX message

```
0x01
```

general HEX message

```
0x04
```

general HEX message converted to binary

```
0B00000100
```

Binary converted to statuses (LSB)

```
0 : RFU - n/a
0 : RFU - n/a
1 : usage_detected - true
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

counter_instant HEX message

```
0x68010000
```

HEX message flip for MSB

```
0x00000168
```

HEX message converted to decimal

```
360 (Liters)
```

fPort 49 Configuration Request Message

Reporting configuration request

Byte 0	Operation
0x00	*reporting_config_request*

Message sample

Message goal: Request reporting configuration

Header

Select Header HEX code

```
0x00
```

Compile message for sending (HEX)

```
0x00
```

Control value in base64 to control after sending

```
AA==
```

Response

Sent to fPort/type 49 in the same format as *reportingr_config_packet*

Metering configuration request

Byte 0	Operation
0x05	*metering_config_request*

Message sample

Message goal: Request metering configuration

Header

Select Header HEX code

```
0x05
```

Compile message for sending (HEX)

```
0x05
```

Control value in base64 to control after sending

```
BQ==
```

Response

Sent to fPort/type 49 in the same format as *metering_config_packet*

Meta position configuration request

Byte 0	Operation
0x10	*meta_pos_config_request*

Message sample

Message goal: Request meta position configuration

Header

Select Header HEX code

```
0x10
```

Compile message for sending (HEX)

```
0x10
```

Control value in base64 to control after sending

```
EA==
```

Response

Sent to fPort/type 49 in the same format as *meta_pos_config_packet*

Meta EIC configuration request

Byte 0	Operation
0x11	*meta_eic_config_request*

Message sample

Message goal: Request meta EIC configuration

Header

Select Header HEX code

```
0x11
```

Compile message for sending (HEX)

```
0x11
```

Control value in base64 to control after sending

```
EQ==
```

Response

Sent to fPort/type 49 in the same format as *meta_eic_config_packet*

fPort 50 Configuration Message

reporting_config_packet

Byte 0	Byte 1	Byte 2..3	Byte 4..5	Byte 6
packet_type	*configured_parameters*	*usage_interval*	*status_interval*	*behaviour*
0x00 *reporting_config_packet*	See fig. 1	type: uint16 unit: min when: *usage_interval* = *configured* 0x00 - disable min - 10min max - 24h default - 1h	type: uint16 unit: min when: *usage_interval* = *configured* min - 1h max - 7days default - 24h	See fig. 2

1.

Bit #	Parameter	Value
0	*usage_interval*	0 - *not_configured* 1 - *configured*
1	*status_interval*	
2	*behaviour*	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

2.

Bit #	Parameter	Value
0	*send_usage*	0 - *only_when_new_data* 1 - *always* *default - 1*
1	RFU	
2	RFU	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Message sample

Message goal: Set usage interval to 120 minutes and behaviour to only when new data

Header

Select Header HEX code

```
0x00
```

configured_parameters

```
1 : usage_interval - configured
0 : status_interval - not_configured
1 : behaviour - configured
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

Selection converted to binary

```
0B00000101
```

Selection converted to HEX

```
0x05
```

usage_interval decimal value

```
120
```

Decimal value converted to HEX

```
0x0078
```

HEX message flip for LSB

```
0x7800
```

behaviour

```
0 : send_usage - only_when_new_data
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

Selection converted to binary

```
0B00000000
```

Selection converted to HEX

```
0x00
```

Compile message for sending (HEX)

```
00|05|7800|00
```

Control value in base64 to control after sending

```
AAV4AAA=
```

metering_config_packet

Byte 0	Byte 1	Byte 2	Byte 3..6	Byte 3..4	Byte 7..8
packet_type	*configured_parameters*	*general_config*	*absolute_reading*	*offset*	*meter_serial*
0x05 *metering_config_packet*	See fig. 1	See fig. 2	type: uint32 unit: l when: *reading_absolute* = *configured*	type: int16 unit: l when: *reading_offset* = *configured*	type: hex when: *meter_serial* = *configured* 0xFFFFFFFF - not_set

1.

Bit #	Parameter	Value
0	*general_config*	0 - *not_configured* 1 - *configured*
1	*reading_absolute**	
2	*reading_offset**	
3	RFU	
4	*meter_serial*	0 - *not_configured* 1 - *configured*
5	RFU	
6	RFU	
7	RFU	

2.

Bit #	Parameter	Value
0	*exponent* (units per pulse)	0 - *ignore* 1 - *1_L* 2 - *10_L* 3 - *100_L* 4 - *1000_L*
1		
2	RFU	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

* *reading_absolute* and *reading_offset* can not be configured together.

Message sample

Message goal: Set reading value to 23.650m³, exponent to 10L and meter serial to 34500027

Header

Select Header HEX code

```
0x05
```

configured_parameters

```
1 : general_config - configured
1 : reading_absolute - configured
0 : reading_offset - not_configured
0 : RFU - n/a
1 : meter_serial - configured
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

Selection converted to binary

```
0B00010011
```

Selection converted to HEX

```
0x13
```

general_config selection

```
x : exponent
x :
x :
0 : RFU
0 : RFU
0 : RFU
0 : RFU
0 : RFU
```

exponent time binary value

```
0B010 (must be 2)
```

Whole binary message assembled

```
0B00000010
```

Binary value converted to HEX

```
0x02
```

absolute_reading decimal value

```
23650
```

Decimal value converted to HEX

```
0x00005C62
```

HEX message flip for LSB

```
0x625C0000
```

meter_serial decimal value

```
34500027
```

Decimal value converted to HEX

```
0x020E6DBB
```

HEX message flip for LSB

```
0xBB6D0E02
```

Compile message for sending (HEX)

```
05|12|02|625C0000|BB6D0E02
```

Control value in base64 to control after sending

```
BRJ7XAAAu20OAg==
```

meta_pos_config_packet

Byte 0	Byte 1	Byte 2..5	Byte 6..9	Byte 10	Byte 11..x
packet_type	*configured_parameters*	*gps_position*		*address*	
		latitude	*longitude*	*length*	*data[0]..data[length-1]*
0x10 *meta_pos_config_packet*	See fig. 1	type: int32 value: $*10^7$ when: *gps_position* = *configured* 0x7FFFFFFF - not_configured	type: int32 value: $*10^7$ when: *gps_position* = *configured* 0x7FFFFFFF - not_configured	type: uint8 unit: B when: *address* = *configured* max=38	type: char array (utf-8) when: *address* = *configured*

1.

Bit #	Parameter	Value
0	*gps_position*	0 - *not_configured* 1 - *configured*
1	*address*	
2	RFU	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Message sample

Message goal: Set the device coordinates to 59.4286400, 24.6610052

Header

Select Header HEX code

```
0x10
```

configured_parameters

```
1 : gps_position - configured
0 : address - not_configured
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

Selection converted to binary

```
0B00000001
```

Selection converted to HEX

```
0x01
```

gps_position

latitude decimal value (7 decimal places)

```
59.4286400
```

Decimal value multipleid by 10000000

```
594286400
```

Decimal value converted to HEX

```
0x236C1740
```

HEX message flip for LSB

```
0x40176C23
```

longitude decimal value (7 decimal places)

```
24.6610052
```

Decimal value multipleid by 10000000

```
246610052
```

Decimal value converted to HEX

```
0x0EB2F884
```

HEX message flip for LSB

```
0x84F8B20E
```

Compile message for sending (HEX)

```
1001|40176C23|84F8B20E
```

Control value in base64 to control after sending

```
EAFAF2wjhPiyDg==
```

meta_eic_config_packet

Byte 0	Byte 1	Byte 2..5	Byte 6..9	Byte 10	Byte 11..x
packet_type	*configured_parameters*	*eic_customer_len*	*eic_customer* *id_1[0]..id_1[eic_customer_len-1]*	*eic_location_len*	*eic_location* *id_1[0]..id_1[eic_location_len-1]*
0x11 *meta_eic_config_packet*	See fig. 1	type: uint8 unit: B when: *eic_customer* = *configured* min=1 max=16	type: char array (utf-8) when: *eic_customer* = *configured*	type: uint8 unit: B when: *eic_location* = *configured* min=1 max=16	type: char array (utf-8) when: *eic_location* = *configured*

1.

Bit #	Parameter	Value
0	*eic_customer*	0 - *not_configured* 1 - *configured*
1	*eic_location*	
2	RFU	
3	RFU	
4	RFU	
5	RFU	
6	RFU	
7	RFU	

Message sample

Message goal: Set the device customer EIC to ee067398900009fi

Header

Select Header HEX code

```
0x11
```

configured_parameters

```
1 : eic_customer - configured
0 : eic_location - not_configured
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

Selection converted to binary

```
0B00000001
```

Selection converted to HEX

```
0x01
```

eic_customer_len decimal value

```
16
```

Decimal value converted to HEX

```
0x10
```

eic_customer

id_1[0..15] char array message (UTF-8)

```
ee067398900009fi
```

char array coverted to HEX

```
65653036373339383930303030396669
```

Compile message for sending (HEX)

```
11|01|16|65653036373339383930303030396669
```

Control value in base64 to control after sending

```
EQEWZWUwNjczOTg5MDAwMDlmaQ==
```

fPort 51 Update message

Byte 0
Header
FF

Activate update mode for BT update for 2 minutes. if nothing is done the device will reboot, join and resume working

NB! **Only** unconfirmed messages should be used for this message.

Message sample

Message goal: Set device to update mode

Header

Select Header HEX code

```
FF
```

Compile message for sending (HEX)

```
FF
```

Control value in base64 to control after sending

```
/w==
```

fPort 99 System Messages

boot_packet

Byte 0	Byte 1..4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11..12	Byte 13..16
packet_type	*device_serial*	*device_fw_version*			*reset_info*	*general_info*	*hardware_config*	*sensor_fw_version*	*device_uptime*
		major	*minor*	*patch*					
0x00 *boot_packet*	type: uint:32 format: HEX	type: uint8	type: uint8	type: uint8	See fig. 1	See fig. 2	0x20 *BK-G*	type: uint16 0x0000 - n/a	type: uint24 unit: h

1.

Bit #	Parameter	Value
0	RFU	
1	*watchdog_reset*	0 - *false* 1 - *true*
2	*soft_reset* (e.g DFU)	
3	RFU	
4	*magnet_wakeup*	0 - *false* 1 - *true*
5	RFU	
6	RFU	
7	*nfc_wakeup*	0 - *false* 1 - *true*

2.

Bit #	Parameter	Value
0	RFU	
1	RFU	
2	RFU	
3	RFU	
4	RFU	
5	*wakeup_from_dfu*	0 - *false* 1 - *true*
6	*rejoined*	
7	*configuration_restored*	

Message sample

Message in base64

```
APoBEFABAgaAACAAAAAAAA==
```

Message decoded to HEX

```
00FA011050010206800020000000000
```

packet_type HEX message

```
0x00
```

HEX message translated to *packet_type*

```
boot_packet
```

device_serial HEX message

```
0xFA011050
```

HEX message flip for MSB

```
0x501001FA
```

device_fw_version

major HEX message

```
0x01
```

HEX message converted to decimal

```
1
```

minor HEX message

```
0x02
```

HEX message converted to decimal

```
2
```

patch HEX message

```
0x06
```

HEX message converted to decimal

```
6
```

reset_info HEX message

```
0x80
```

HEX message converted to binary

```
0B10000000
```

Binary converted to configuration (LSB)

```
0 : RFU - n/a
0 : watchdog_reset - false
0 : soft_reset - false
0 : RFU - n/a
0 : magnet_wakeup - false
0 : RFU - n/a
0 : RFU - n/a
1 : nfc_wakeup - true
```

general_info HEX message

```
0x00
```

HEX message converted to binary

```
0B00000000
```

Binary converted to configuration (LSB)

```
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : wakeup_from_dfu - false
0 : rejoined - false
0 : configuration_restored - false
```

hardware_config HEX message

```
0x20
```

HEX message translated to *hardware_config*

```
BK-G
```

sensor_fw_version HEX message

```
0x0000
```

HEX message flip for MSB

```
0x0000
```

HEX message converted to decimal

```
0 (n/a)
```

device_uptime HEX message

```
0x000000
```

HEX message flip for MSB

```
0x000000
```

HEX message converted to decimal

```
0 (hours)
```

shutdown_packet

Byte 0	Byte 1	Byte 2..x
packet_type	*shutdown_reason*	*last_status_packet*
0x01 *shutdown_packet*	See fig. 1	Value: full payload of regular *status_message*

1.

Value	reason
0x10	*calibration_timeout*
0x20	*hardware_error*
0x31	*magnet_shutdown*
0x32	*enter_dfu*
0x33	*app_shutdown*
0x34	*switch_to_wmbus*

Message sample

Message in base64

```
ATIBAAD/ERAATwcBAAAAAAAA
```

Message decoded to HEX

```
0132010000FF1110004F0701000000000000
```

packet_type HEX message

```
0x01
```

HEX message translated to *packet_type*

```
shutdown_packet
```

shutdown_reason HEX message

```
0x32
```

HEX message converted to *shutdown_reason*

```
enter_dfu
```

status_packet

header HEX message

```
0x01
```

general HEX message

```
0x00
```

general HEX message converted to binary

```
0B00000000
```

Binary converted to statuses (LSB)

```
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : packet_reason_app - false
0 : packet_reason_magnet - false
0 : packet_reason_alert - false
```

active_alerts HEX message

```
0x00
```

active_alerts HEX message converted to binary

```
0B00000000
```

```
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : tamper - false
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
0 : RFU - n/a
```

battery

percentage HEX message

```
0xFF
```

HEX message converted to value

```
not_available
```

voltage HEX message

```
0x92
```

HEX message converted to decimal

```
146
```

Decimal value mapped to voltage

```
3.246 (V)
```

board

temperature HEX message

```
0x12
```

HEX message converted to signed decimal

```
18 (°C)
```

extremes HEX message

```
0x00
```

extremes HEX message converted to binary

```
0B00000000
```

Binary converted to extremes (LSB)

```
0 : min_offset [0B0000]
0 :
0 :
0 :
0 : max_offset [0B0000]
0 :
0 :
0 :
```

min_offset binary value converted to HEX

```
0x00
```

HEX message converted to decimal

```
0
```

Decimal offset value multiplied by -2

```
0 (°C)
```

max_offset binary value converted to HEX

```
0x00
```

HEX message converted to decimal

```
0
```

Decimal offset value multiplied by 2

```
0 (°C)
```

downlink

rssi HEX message

```
0x52
```

HEX message converted to decimal

```
82
```

Decimal value multiplied by -1

```
-82 (dBm)
```

snr HEX message

```
0x07
```

HEX message converted to signed decimal

```
7 (dB)
```

counter_instant HEX message

```
0xA0000000
```

HEX message flip for MSB

```
0x000000A0
```

HEX message converted to decimal

```
160 (Liters)
```

config_failed_packet

Byte 0	Byte 1	Byte 2..x
packet_type	*packet_from_fport*	*parse_error_code*
0x13 *config_failed_packet*	type: uint8	See fig. 1

1.

Value	reason
2	*unknown_fport*
3	*packet_size_short*
4	*packet_size_long*
5	*value_error*
6	*protocol_parse_error*
7	*reserved_flag_set*
8	*invalid_flag_combination*
9	*unavailable_feature_request*
10	*unsupported_header*
11	*unreachable_hw_request*
13	*internal_error*

Message sample

Message in base64

```
EzMK
```

Message decoded to HEX

```
13330A
```

packet_type HEX message

```
0x13
```

HEX message translated to *packet_type*

```
config_failed_packet
```

packet_from_fport HEX message

```
0x33
```

HEX message converted to decimal

```
(fPort) 51
```

parse_error_code HEX message

```
0x0A
```

HEX message converted to decimal

```
10
```

Decimal message converted to error

```
unsupported_header
```

wM-Bus Message

Byte #	Payload (HEX)	Decrypted Payload (HEX)	Block information	Field Information
0	1E	not encrypted	wM-Bus Header	L-field. 30 Bytes to follow
1	44			C-field. SND-NR (Send/No Reply)
2	33			Manufacturer ID. 0x3833 > 0b01110 0b00001 0b10011 > NAS
3	38			
4	46			A-Field Device ID. Little Endian BCD 20040646
5	06			
6	04			
7	20			
8	01			A-field Version
9	03			A-field Type. 0x03 - Gas meter
10	7A		Payload Header	Short header
11	5A			ACC - Access number
12	00			Status byte
13	10			Configuration field. Value 0x2510 Bits: BASMMMMMNNNNCCRH S - 1(synchronous message) M - 5(OMS Security Mode 5 N - 1(Nr of encryption blocks)
14	25			
15	8C	2F	Encrypted Application Payload	Encryption verification
16	CB	2F		
17	50	02		DIF - 16bit integer, OMS 4.1.2 Data point ID6!, unique message identification
18	B6	FD		VIF - linear VIF extension, next byte is VIF
19	2B	08		VIFE - Unique message identifcation
20	C3	B8		0x18B8 > tx count = 6328
21	66	18		
22	E0	04		DIF - 32 bit integer, OMS 4.1.2 Data point VM1!, Volume, current value, total
23	DF	13		VIF - Volume, unit liters
24	E7	2C		0x0000012C > 300 liters
25	57	01		
26	41	00		
27	53	00		
28	95	2F		Encryption padding
29	92	2F		
30	3d	2F		

3.6V BATTERY OFFSET CHART

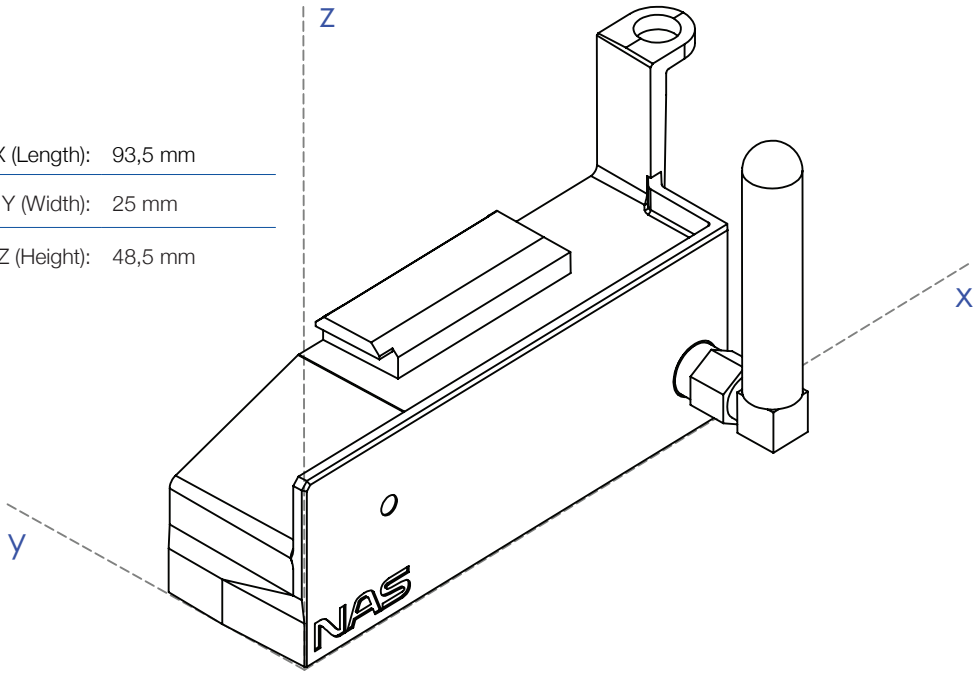
255 - Not measured

254 - 4
253 - 3,95
252 - 3,9
251 - 3,85
250 - 3,8
249 - 3,75
248 - 3,7
247 - 3,65
246 - 3,646
245 - 3,642
244 - 3,638
243 - 3,634
242 - 3,63
241 - 3,626
240 - 3,622
239 - 3,618
238 - 3,614
237 - 3,61
236 - 3,606
235 - 3,602
236 - 3,606
235 - 3,602
234 - 3,598
233 - 3,594
232 - 3,59
231 - 3,586
230 - 3,582
229 - 3,578
228 - 3,574
227 - 3,57
226 - 3,566
225 - 3,562
224 - 3,558
223 - 3,554
222 - 3,55
221 - 3,546
220 - 3,542
219 - 3,538
218 - 3,534
217 - 3,53
216 - 3,526
215 - 3,522
214 - 3,518
213 - 3,514
212 - 3,51
211 - 3,506
210 - 3,502
209 - 3,498
208 - 3,494
207 - 3,49
206 - 3,486
205 - 3,482
204 - 3,478
203 - 3,474
202 - 3,47
201 - 3,466
200 - 3,462
199 - 3,458
198 - 3,454
197 - 3,45
196 - 3,446
195 - 3,442
194 - 3,438
193 - 3,434
192 - 3,43
191 - 3,426
190 - 3,422
189 - 3,418
188 - 3,414
187 - 3,41
186 - 3,406
185 - 3,402
184 - 3,398
183 - 3,394
182 - 3,39
181 - 3,386
180 - 3,382
179 - 3,378
178 - 3,374
177 - 3,37
176 - 3,366
175 - 3,362
174 - 3,358
173 - 3,354
172 - 3,35
171 - 3,346
170 - 3,342
169 - 3,338
168 - 3,334
167 - 3,33
166 - 3,326
165 - 3,322
164 - 3,318
163 - 3,314
162 - 3,31
161 - 3,306
160 - 3,302
159 - 3,298
158 - 3,294
157 - 3,29
156 - 3,286
155 - 3,282
154 - 3,278
153 - 3,274
152 - 3,27
151 - 3,266
150 - 3,262
149 - 3,258
148 - 3,254
147 - 3,25
146 - 3,246
145 - 3,242
144 - 3,238
143 - 3,234
142 - 3,23
141 - 3,226
140 - 3,222
139 - 3,218
138 - 3,214
137 - 3,21
136 - 3,206
135 - 3,202
134 - 3,198
133 - 3,194
132 - 3,19
131 - 3,186
130 - 3,182
129 - 3,178
128 - 3,174
127 - 3,17
126 - 3,166
125 - 3,162
124 - 3,158
123 - 3,154
122 - 3,15
121 - 3,146
120 - 3,142
119 - 3,138
118 - 3,134
117 - 3,13
116 - 3,126
115 - 3,122
114 - 3,118
113 - 3,114
112 - 3,11
111 - 3,106
110 - 3,102
109 - 3,098
108 - 3,094
107 - 3,09
106 - 3,086
105 - 3,082
104 - 3,078
103 - 3,074
102 - 3,07
101 - 3,066
100 - 3,062
99 - 3,058
98 - 3,054
97 - 3,05
96 - 3,046
95 - 3,042
94 - 3,038
93 - 3,034
92 - 3,03
91 - 3,026
90 - 3,022
89 - 3,018
88 - 3,014
87 - 3,01
86 - 3,006
85 - 3,002
84 - 2,998
83 - 2,994
82 - 2,99
81 - 2,986
80 - 2,982
79 - 2,978
78 - 2,974
77 - 2,97
76 - 2,966
75 - 2,962
74 - 2,958
73 - 2,954
72 - 2,95
71 - 2,946
70 - 2,942
69 - 2,938
68 - 2,934
67 - 2,93
66 - 2,926
65 - 2,922
64 - 2,918
63 - 2,914
62 - 2,91
61 - 2,906
60 - 2,902
59 - 2,898
58 - 2,894
57 - 2,89
56 - 2,886
55 - 2,882
54 - 2,878
53 - 2,874
52 - 2,87
51 - 2,866
50 - 2,862
49 - 2,858
48 - 2,854
47 - 2,85
46 - 2,846
45 - 2,842
44 - 2,838
43 - 2,834
42 - 2,83
41 - 2,826
40 - 2,822
39 - 2,818
38 - 2,814
37 - 2,81
36 - 2,806
35 - 2,802
34 - 2,798
33 - 2,794
32 - 2,79
31 - 2,786
30 - 2,782
29 - 2,778
28 - 2,774
27 - 2,77
26 - 2,766
25 - 2,762
24 - 2,758
23 - 2,754
22 - 2,75
21 - 2,746
20 - 2,742
19 - 2,738
18 - 2,734
17 - 2,684
16 - 2,634
15 - 2,584
14 - 2,534
13 - 2,484
12 - 2,434
11 - 2,384
10 - 2,334
9 - 2,284
8 - 2,234
7 - 2,184
6 - 2,134
5 - 2,084
4 - 2,034
3 - 1,984
2 - 1,934
1 - 1,884

0 - N/A

DIMENSIONS

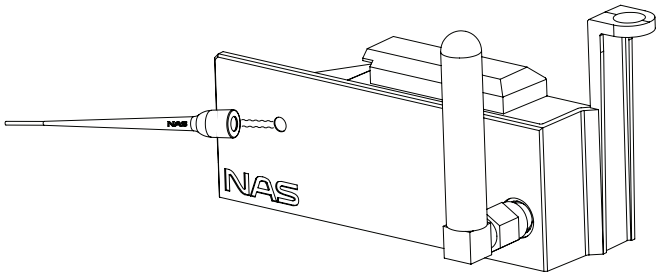
X (Length):	93,5 mm
Y (Width):	25 mm
Z (Height):	48,5 mm



ACTIVATION

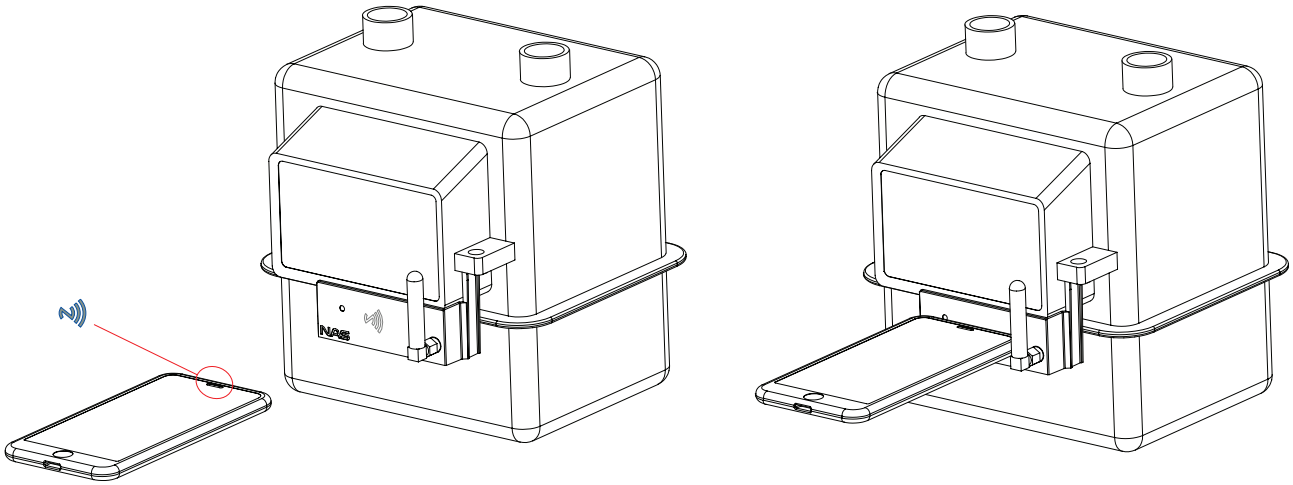
Magnet Activation

To activate the device, hold the magnet against the device approximately on the NAS logo (see illustration) for atleast 1 second. You can remove the magnet when the red light starts to flash

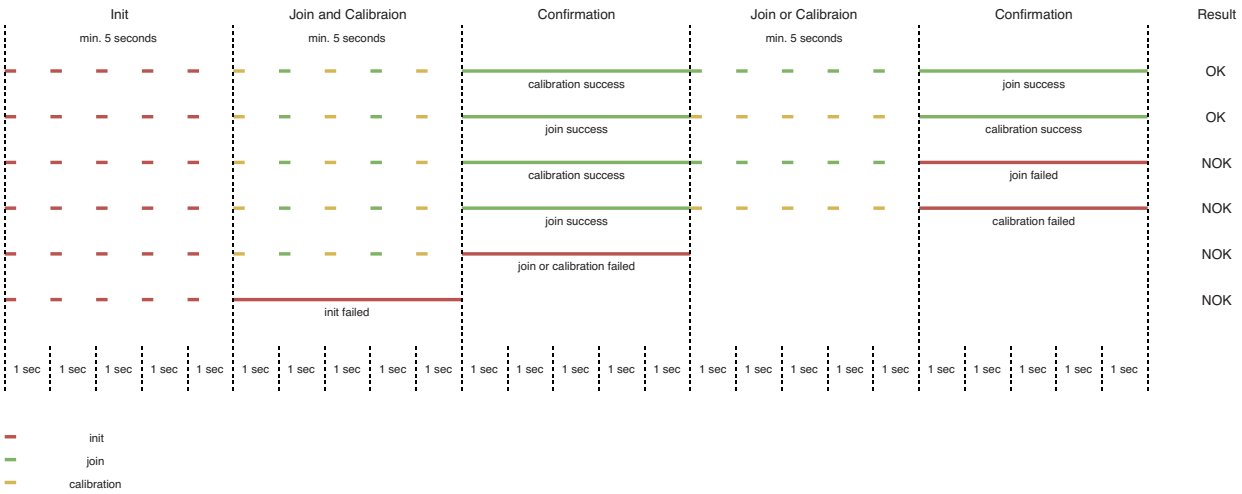


NFC Activation

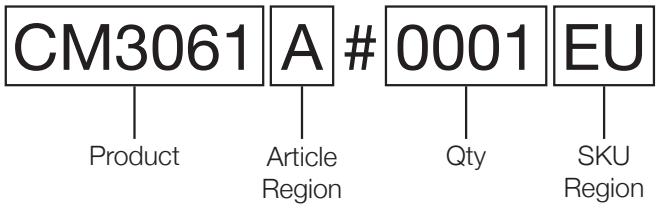
To activate the device, run the NAS Connect application on your phone and scan the NFC on the device (see illustration). You can remove the phone when the “Check” icon is displayed on the phone.



LED behaviour during activation



ORDERING INFORMATION



Product/SKU	Package qty	Version
CM3061x#0001xx	1	BK-G

Article region	SKU region	Band
A	EU	EU868
B	AU	AU915
C	US	US915
D	AS	AS923
F	KR	KR920
I	IN	IN865
J	RU	RU864

CONTACT INFORMATION

Nordic Automation Systems AS

www.nasys.no

info@nasys.no

REVISION HISTORY

1.0 - First version

1.1 - Support for FW 0.8.4

- Boot packet battery info removed, config restored added
- Status packet optional active_alerts Byte, added battery % byte, added temperature extremes byte. Added device serial option for digital interface
- Configuration packet new multiplyer logic. Possible to offset reading instead of only setting. Possible to add device serial.
- Configuration request packet has separate request for reporting and gas config.

2.0 - Support for FW 1.2.1

- Boot packet: device uptime added
- wM-Bus support
- Status message: downlink snr added, message optimised
- Usage message: optimised
- Configuration request: Metering configuration unified, offset reworked, EIC configuration added
- Configuration request: Configurations separated to align with configuration packets