atim cloud wireless™
PRODUCT LINE

ATIM Downlink Protocol

ACW-DINRSM Remote Configuration

Technical documentation

ATIM Radiocommunications
Chemin des Guillets
38250 Villard de Lans
France

www.atim.com
info@atim.com
+33 4 76 95 50 65

1 Table of Contents

2 Version history of this document:

Document version	Date	Description	Author
1.0	12/03/2025	First version	YLB
1.1	19/05/2025	Fixed the "size" field in the configuration frames ofmodbus 1-49 queries	YLB

3 Description

This document describes the list of modifiable settings for the product as well as the commands available on it. Product re-parameterization and orders are transmitted to the means of downlinks

There are two types of downlinks possible:

- Configuration downlinks (see Configuration Frame). They allow you to change the settings of the product.

Command downlinks (see The "**Thr_low**" and "**Thr_high**" fields are used to define the thresholds. They are encoded on 2 bytes with the format little endian

IMPORTANT

The first configured modbus request corresponds to the modbus parameter 1 (header 0xCF), the second corresponds to the modbus parameter 2 (rheader 0xD0) and so on.

*As soon as a modbus parameter has its "**code**" field at 0 (No Modbus request), all subsequent parameters will not be processed by the product, even if they contain valid modbus requests ("code" field other than 0)*

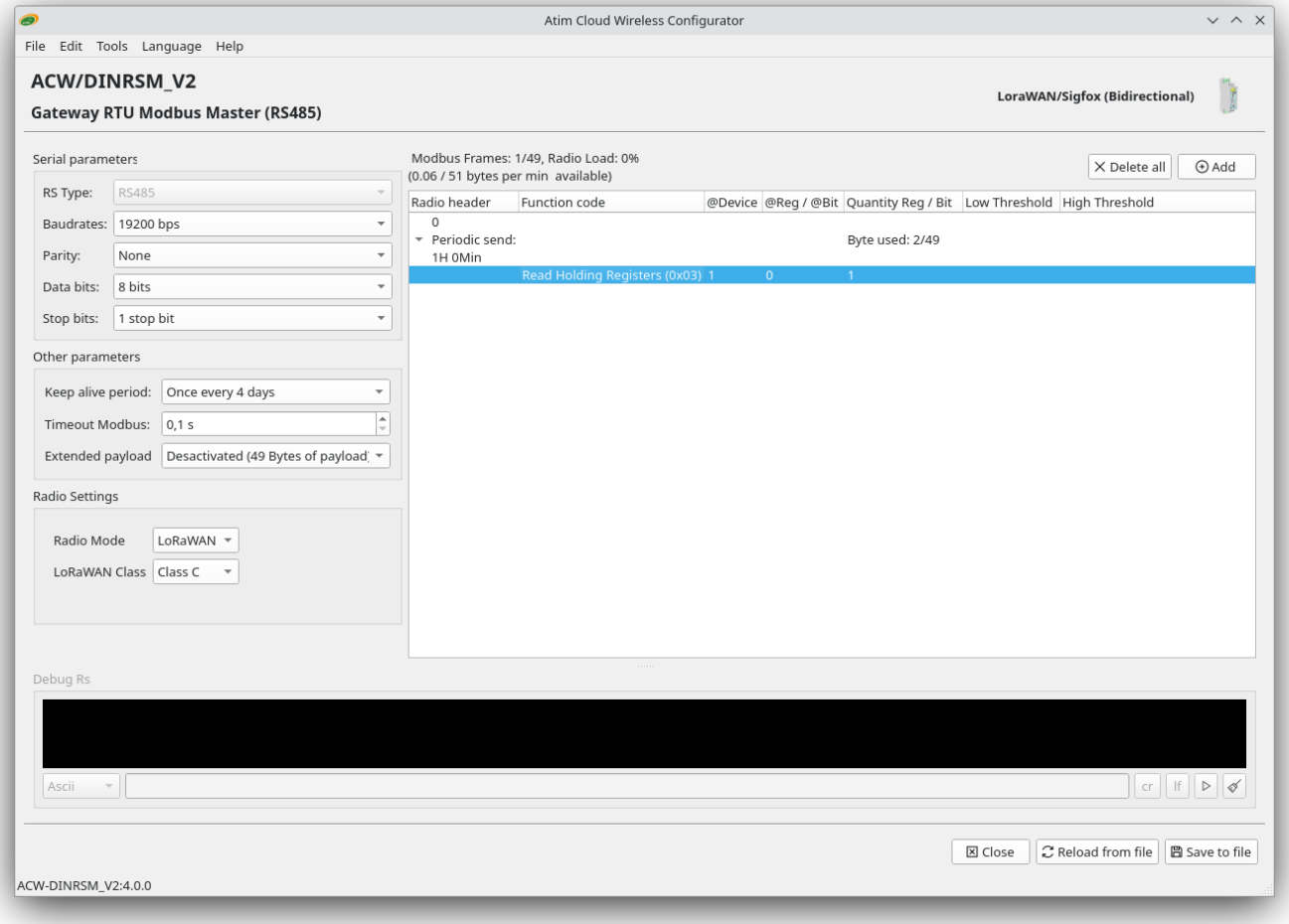
That is to say, it is not possible to configure several queries on discontinuous parameters. It must necessarily have continuity in the settings when configuring the product.

3.1.1.1 Example of configuration downlinks

3.1.1.1.1 Simple Setup Sending Case

Case with a single periodic modbus request:

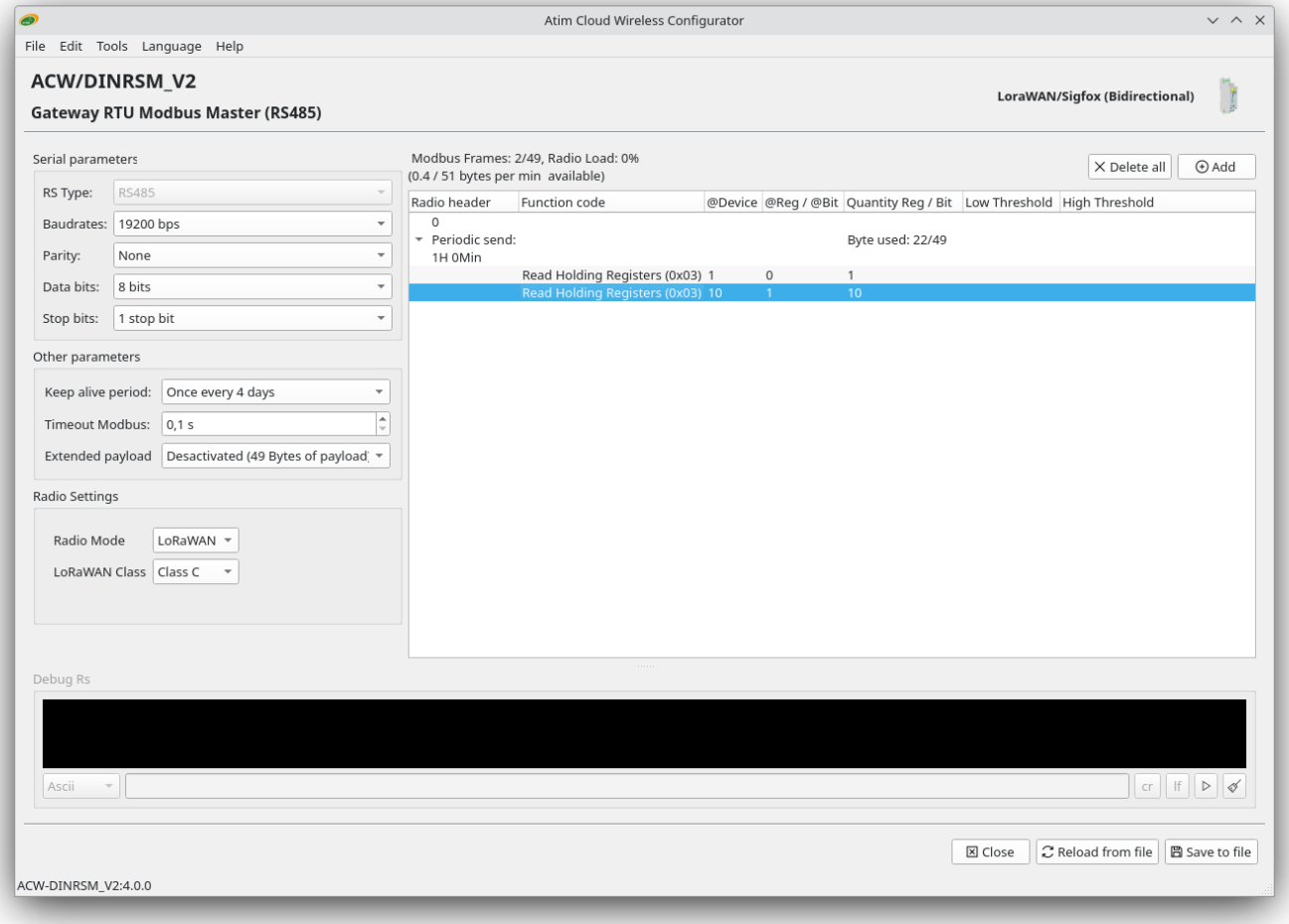
- Period 1h (header frame 0)
- request to read register 0x03
- Modbus slave address 1
- reading 1 register from register 0



=> in this case you have to send the downlink :
CF11000301000000010001000000000000000

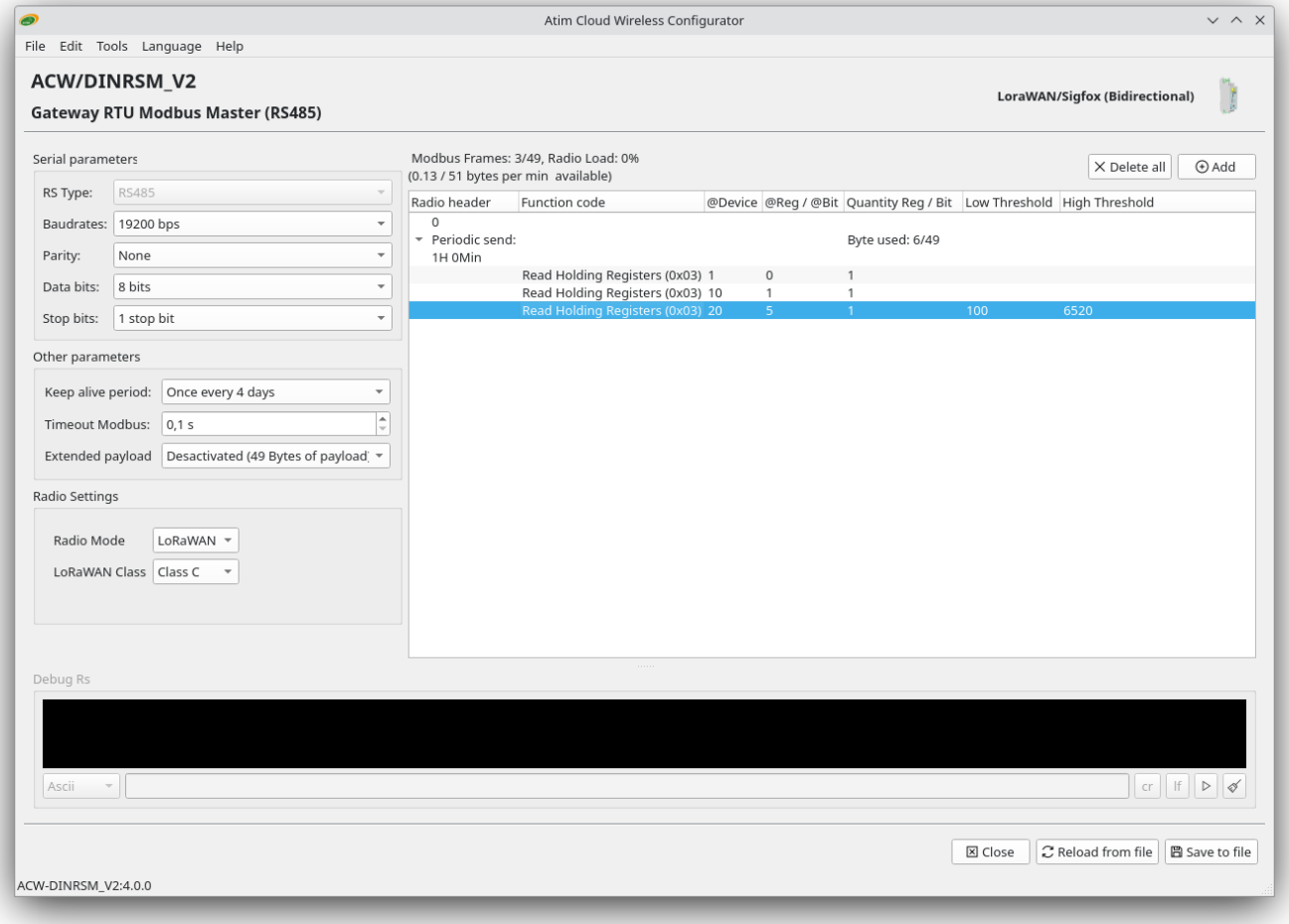
If we add a query in the same radio frame with:

- request to read register 0x04
- Modbus slave address 10
- Reading 5 registers from register 1



The configuration downlink for the second modbus request to be sent is:
D01100830A0100000A00010000000000000000

With an additional query in the radio frame (with a high and low threshold configured):



The configuration downlink of the 3rd modbus request is:

D11100831405000001000100000000364007819

NOTE

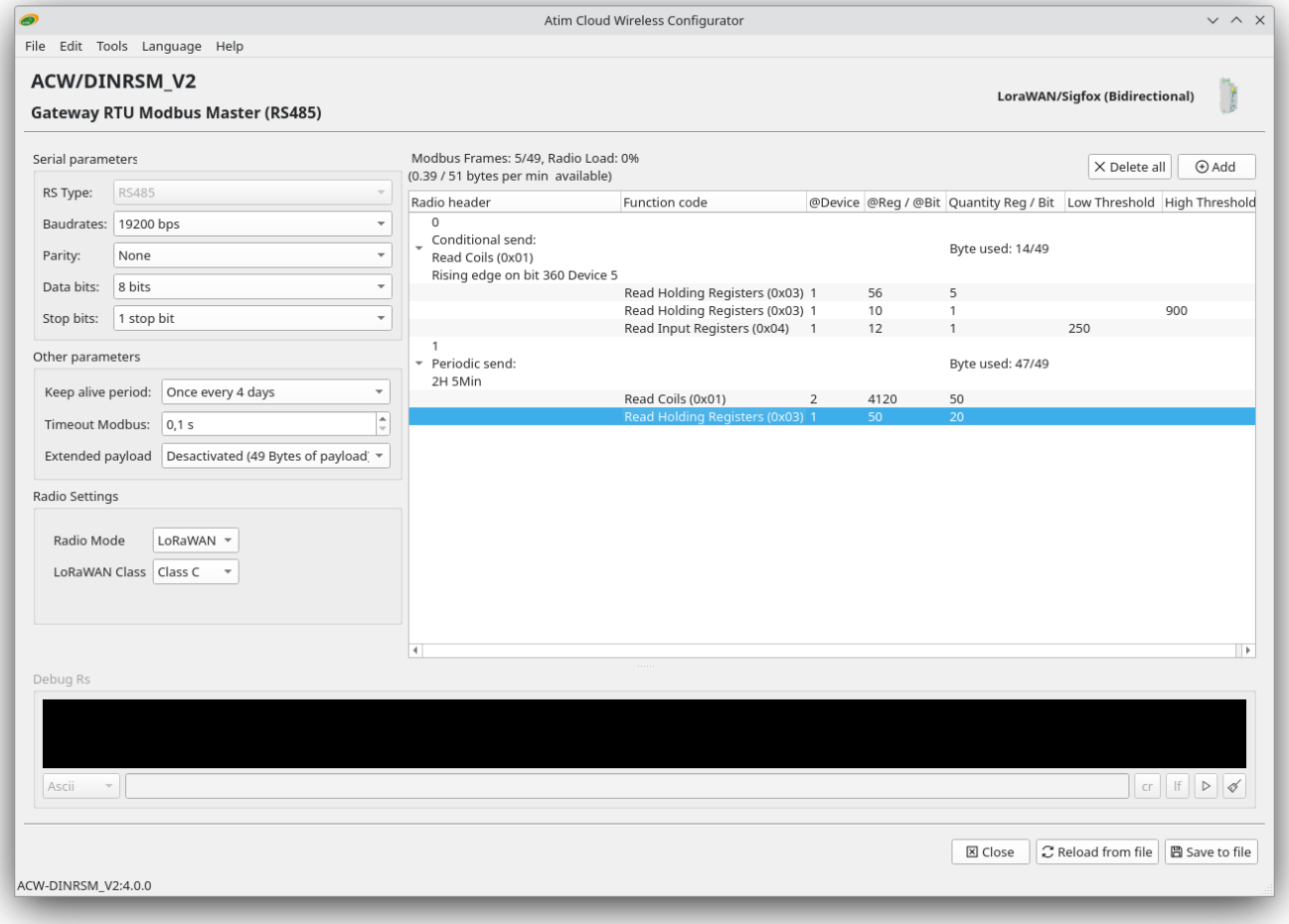
It is possible to concatenate downlinks into a single downlink, up to a limit of 51 bytes. In this example, if we want to minimize the number of downlinks to be sent in the case of a configuration of these 3 requests, we can concatenate them into 2 downlinks:

Downlink 1:
CF1100030100000001000100000BA0000000000D01100830A0100000100010000 6E0000000000D10B00831405000001000100000

Downlink 2:
D1070A6E0364007819

Request 3 is split into two (one part in the 1st downlink and one part in the second downlink). This is possible because all "modbus request" parameters are buffered parameters.

3.1.1.1.2 Cases with multiple radio frames and multiple queries per frame



In this configuration, there are 5 modbus requests spread over 2 radio frames.

Downlink query parameter 1: **CF11000301380000050105056801000000000Downlink query parameter 2: D0110083010A0000010105056801020000 8403Downlink query parameter 3: D1110084010C0000010105056801 01FA000000Downlink query parameter 4: D2110001021810013200020500650000000000Downlink query parameter 5: D3110083013200011400020500650000000000**

NOTE

It is possible to concatenate downlinks into a single downlink, up to a limit of 51 bytes. In this example, if we want to minimize the number of downlinks to be sent in the case of a configuration of these 5 requests, we can concatenate them into 2 downlinks:

Downlink 1:
CF11000301380000050105056801000000000D0110083010A0000010105056801020000840 3D10B0084010C00000101050568

Downlink 2:
D1070A0101FA000000D2110001021810013200020500650000000000D3110083013200011400020500650000000000

Request 3 is split into two (one part in the 1st downlink and one part in the second downlink). This is possible because all "modbus request" parameters are buffered parameters.

- Control frame). They allow the product to be controlled remotely.

IMPORTANT NOTES :

- *It is possible to send a configuration downlink that contains several parameters to be modified. To do this, you just have to concatenate the downlinks corresponding to the parameters to be modified. The size of the final downlink must not exceed 51 bytes.*
- *It is not possible to send a downlink that contains both a parameter change and a command.*
- *It is not possible to send a downlink containing multiple commands. A command must always be isolated in a single downlink.*

4 Configuration Frame

4.1 Product Parameter List

Parameter	Value (hexa)	waist
Keep Alive	0x03	1
Modbus Serial	0x0A	2
Extended Size	0x0B	1
Modbus Timeout	0x0D	2
Modbus Request 1 (buffered parameter)	0x0F	16 (Extended Frame)
Modbus Request 2 (buffered parameter)	0x10	16 (Extended Frame)
Modbus Request 3 (buffered parameter)	0x11	16 (Extended Frame)
Modbus Request 4 (buffered parameter)	0x12	16 (Extended Frame)
Modbus Request 5 (buffered parameter)	0x13	16 (Extended Frame)
...	...	...
Modbus Request 47 (buffered parameter)	0x3D	16 (Extended Frame)
Modbus Request 48 (buffered parameter)	0x3E	16 (Extended Frame)
Modbus Request 49 (buffered parameter)	0x3F	16 (Extended Frame)

4.1.1 0x03: Keep Alive

Byte 0 - Header	Byte 1
0x03	Period

With the following possible values for the "period" field:

Value	Period
0x04	once every 10 mins
0x05	once every hours
0x0a	once every 2 hours
0x0b	once every 4 hours
0x0c	once every 8 hours
0x06	once every day
0x0d	once every 2 days
0x0e	once every 3 days
0x0f	once every 4 days
0x07	once every week
0x08	once every month (30 days)

EXAMPLE

*To change the Keep Alive period to once a day, you will need to send the downlink **0x0306***

4.1.2 0x0A Modbus Serial

Parameter to configure the modbus serial link

Byte 0 - Header	Byte 1 -2
0x4A (0x0A | 0x40)	Configuration

The "Configuration" parameter is 2 bytes long as described:

Byte 0								Byte 1							
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Baudrate				Reserved		Type		Stop bits		Data bits		Reserved		Parity	

With:

Value	Baudrate
0x00	1200 bps
0x01	2400 bps
0x02	4800 bps
0x03	9600 bps
0x04	19200 bps
0x05	38400 bps
0x06	57600 bps
0x07	115200 bps

Value	Type
0x00	reserved
0x01	RS485

Value	Stop bits
0x00	1 stop bit
0x02	2 stop bits

Value	Data bits

0x00	7-bit data
0x01	8-bit data

Value	Parity
0x00	None
0x01	Odd
0x02	Even

EXAMPLE

To change the serial link setting with:

- 19200 bps

- RS485 type

- 1 stop bit

- 8-bit data

- parity Odd

*=> will you have to send the downlink **0x4A4151***

4.1.3 0x0B Extended size

A setting to extend the maximum size of radio frames from 51 bytes to 115 bytes.

Be careful, if the size is extended to 115 bytes, the product will freeze its Spreading Factor at SF9, which will decrease the radio performance.

Byte 0 - Header	Byte 1
0x0B	Extended

With

Value	Extended
0x00	Non-Jumbo Frame (51 Bytes Max)
0x01	Jumbo Frame (115 Bytes max)

EXAMPLE

*To switch to "extended" mode, you need to send the downlink **0x0B01***

4.1.4 0x0D Modbus Timeout

A setting that sets the maximum wait time for a modbus request before going on timeout.

Byte 0 - Header	Byte 1 -2
0x4D (0x0D | 0x40)	Timeout

With the "timeout" field in ms (little endian)

Min timeout: 10 ms

Max timeout: 65000 ms

EXAMPLE

*To set the modbus timeout to 100ms, you have to send the downlink **0x4D6400***
*To set the modbus timeout to 2s, you have to send the downlink **0x4DD007***

4.1.5 0x0F-0x3F Modbus Request (1 to 49)

These 49 parameters allow you to configure all modbus requests that the product (master) will be able to send to the different modbus devices

Byte 0 - Header	Byte 1 - Size	Byte2 - Index	Bytes
0xCF (0x0F | 0xC0)	Size (0x11)	Index	Request

Table 1 : modbus Request 1

Byte 0 - Header	Byte 1 - Size	Byte2 - Index	Bytes
0xD0 (0x10 | 0xC0)	Size (0x11)	Index	Request

Table 2 : Modbus Request 2

...

Byte 0 - Header	Byte 1 - Size	Byte2 - Index	Bytes
0xFF (0x3F | 0xC0)	Size (0x11)	Index	Request

Table 3 : Modbus request 49

All these "Modbus Request" parameters are buffered parameters, one can modify only one part of the parameters (from the "Index" bytes) if desired.

The "Size" field corresponds to the size of the query from the "Index" byte. In the event that the entire parameter is changed (16-byte "Request" in size), the "Size" field is therefore 0x11 (16 bytes + one byte for the "Index" field).

the "Request" parameter is 16 bytes long as described:

Byte 0								Byte 1	Bytes 2-3	Byte 4	Byte 5
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Slave_addr	register	Radio_header	Size
Concat	Reserved					code					

Byte 6	Bytes 7-10	Byte 11	Bytes 12-13	Bytes 14-15
Req_type	Req_conf	Threshold_conf	Thr_low	Thr_high

Value	Concat
0x00	A query that is not concatenated to the previous one. This means that the result of this query is not part of the same radio frame as the result of the previous query. A new radio frame is therefore created with a value of "0x00"
0x01	A request concatenated to the previous one. This means that the result of this query is part of the same radio frame as the result of the previous query. There is therefore no creation of a new radio frame with a value of "0x01". **Please note: as long as this field is set to "0x01", the fields:** **- Radio_header (byte 4)** **- Req_type (Byte 6)** **- Req_conf (bytes 7-10)** **Must be identical to the fields of the query with which it is concatenated. Indeed, as these 2 requests are part of the same radio frame, the fields linked to this radio frame must be identical.**

Value	Code
0x00	No Modbus request
0x01	Modbus Function Code 0x01 (Read Coils)
0x02	Modbus Function Code 0x02 (Read Discrete Inputs)
0x03	Modbus Function Code 0x03 (Read Holding Registers)
0x04	Modbus Function Code 0x04 (Read Input Registers)

The "**Slave_addr**" field corresponds to the modbus address of the device to which the request is to be sent.

The "**Register**" field corresponds to:

- at the address of the 1st register to be read (in the case of codes 0x03 and 0x04)
- To the address of the 1st bit to be read (in the case of codes 0x01 and 0x02)

The "**Radio_header**" field corresponds to the identifier that will be transmitted in the radio frame containing the response to this modbus request

Please note: in the case of a concatenated query, this field must be identical to the "**Radio_header**" field of the query to which it is attached

The "**Size**" field corresponds to:

- The number of records to be read (in the case of 0x03 and 0x04 codes)
- The number of bits to be read (in the case of 0x01 and 0x02 codes)

Value	Req_type
0x00	Periodic query
0x01	Conditional Motion

The "**Req_conf**" field is used to configure the attributes related to the type of query

4.1.5.1 Req_Conf if periodic query

The "**Req_conf**" field is 4 bytes long:

Byte 0	1 bytes	Byte 2 -3
Period_hour	Period_minute	Reserved

With "**Period_hour**" the "time" part of the modbus request periodicity. With "**Period_minute**" the "minute" part of the periodicity of modbus requests.

EXAMPLE

For a periodic query with a period of 2h05min:

- Periode_hour is "2"

- Period_minute is "5"

4.1.5.2 Req_Conf if conditional request

If the modbus request is of the conditional type, it will not be made periodically, but rather asynchronously, as soon as a configured condition is satisfied.

The condition that triggers the modbus request is a condition on a change in the state of a bit of a modbus slave register.

Periodically (every 15 seconds) the product comes to check if this condition is met. If so, then the modbus query will be executed.

The configuration of the condition is done through the "**Req_conf**" field and is 4 bytes long:

<table>
<tr><th>Byte 0</th><th colspan="8">1 bytes</th><th>Byte 2 -3</th></tr>
<tr><td rowspan="2">Req_dev_addr</td><td>bit7</td><td>bit6</td><td>bit5</td><td>bit4</td><td>bit3</td><td>bit2</td><td>bit1</td><td>bit0</td><td rowspan="2">Req_conf_read_bit (little endian)</td></tr>
<tr><td colspan="4">Reserved</td><td colspan="2">Req_conf_trigger</td><td colspan="2">Req_conf_code</td></tr>
</table>

With the "**Req_dev_addr**" field, the address of the modbus slave.

With:

Value	Req_conf_code
0x01	Modbus Function Code 0x01 (Read Coils)
0x02	Modbus Function Code 0x02 (Read Discrete Inputs)

With:

Value	Req_conf_trigger
0x01	Rising forehead
0x02	Descending Front
0x03	Rising and falling forehead

With the "**Req_conf_read**" field, the address of the bit on which the condition is to be checked

Whatever the type of query, you can also configure high and low thresholds on the reading of a register.

The threshold check takes place every 15 seconds (even in the case of a periodic query). If a threshold is exceeded, an alert radio is then issued.

Attention: The threshold check is only possible in the case of a read request (code field to 0x03 or 0x04) on **one and only one** register ("**Size**" field to 1).

There is no possible threshold on bit read (code field at 0x01 or 0x02), or if the query contains several consecutive registers to be read ("**Size**" field greater than 1).

Value	Threshold_conf
0x00	No threshold verification
0x01	Setting up a low threshold
0x02	Setting up a high threshold

The "**Thr_low**" and "**Thr_high**" fields are used to define the thresholds. They are encoded on 2 bytes with the format little endian

IMPORTANT

The first configured modbus request corresponds to the modbus parameter 1 (header 0xCF), the second corresponds to the modbus parameter 2 (rheader 0xD0) and so on.

*As soon as a modbus parameter has its "**code**" field at 0 (No Modbus request), all subsequent parameters will not be processed by the product, even if they contain valid modbus requests ("code" field other than 0)*

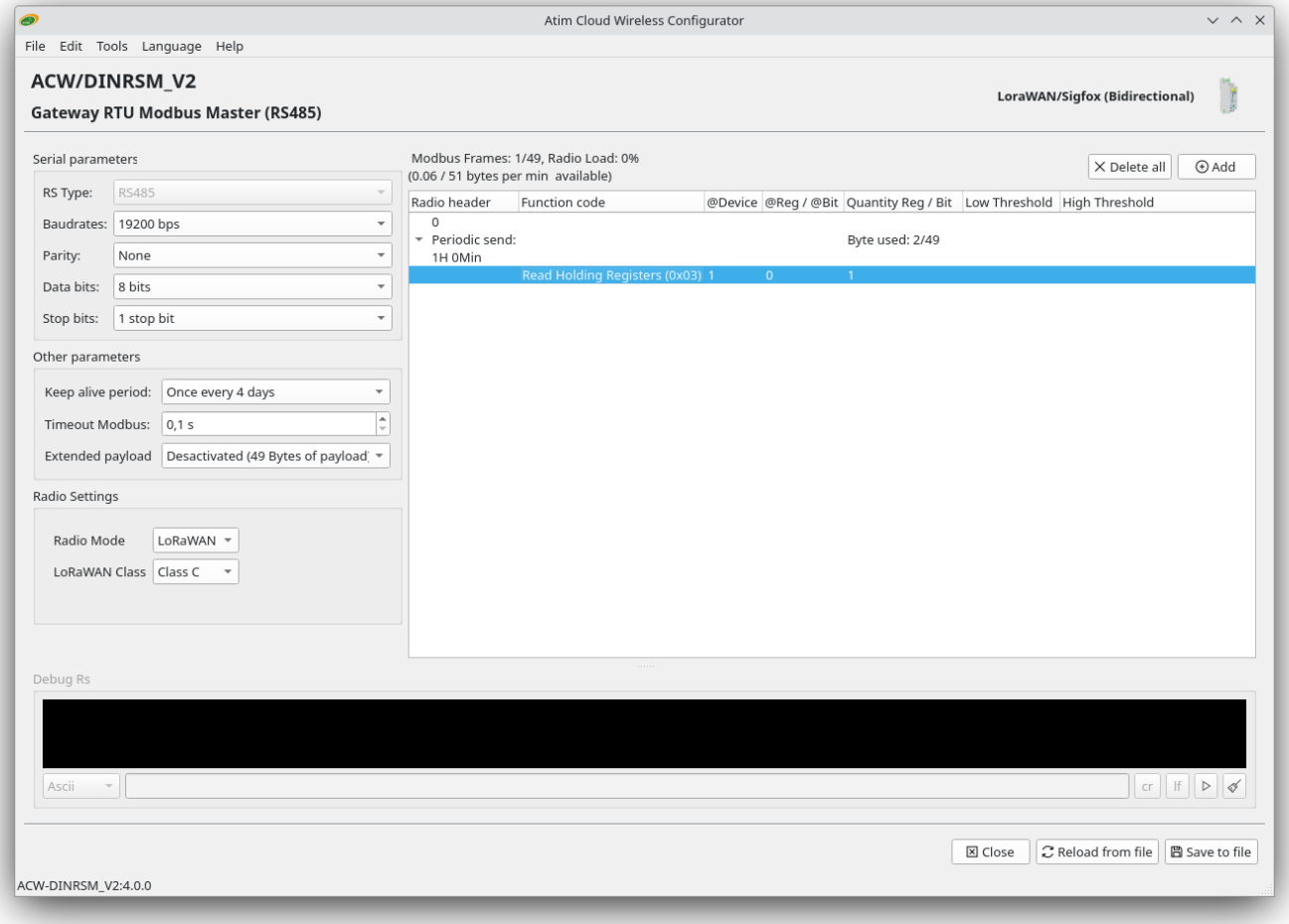
That is to say, it is not possible to configure several queries on discontinuous parameters. It must necessarily have continuity in the settings when configuring the product.

4.1.5.3 Example of configuration downlinks

4.1.5.3.1 Simple Setup Sending Case

Case with a single periodic modbus request:

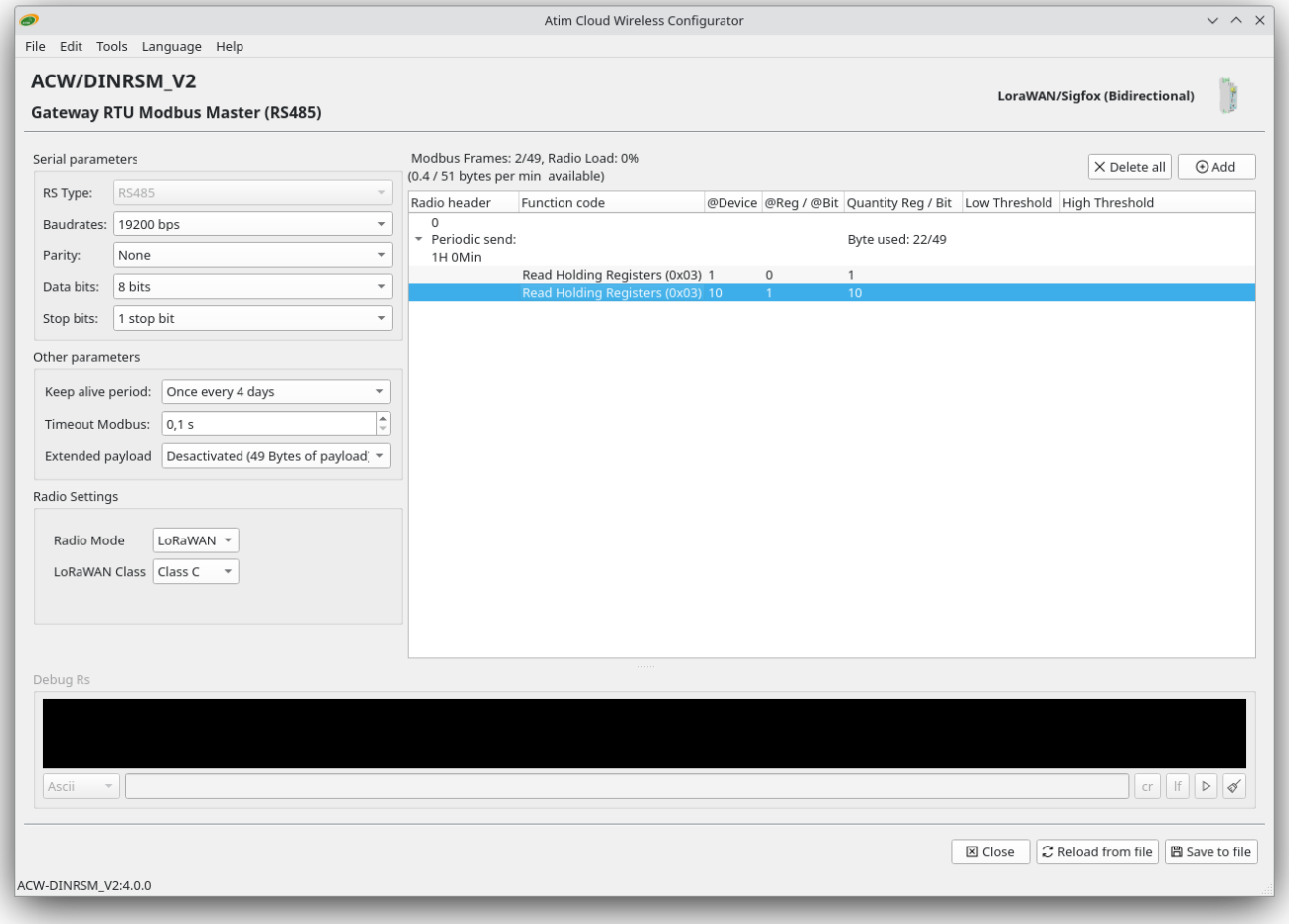
- Period 1h (header frame 0)
- request to read register 0x03
- Modbus slave address 1
- reading 1 register from register 0



=> in this case you have to send the downlink :
CF11000301000000010001000000000000000

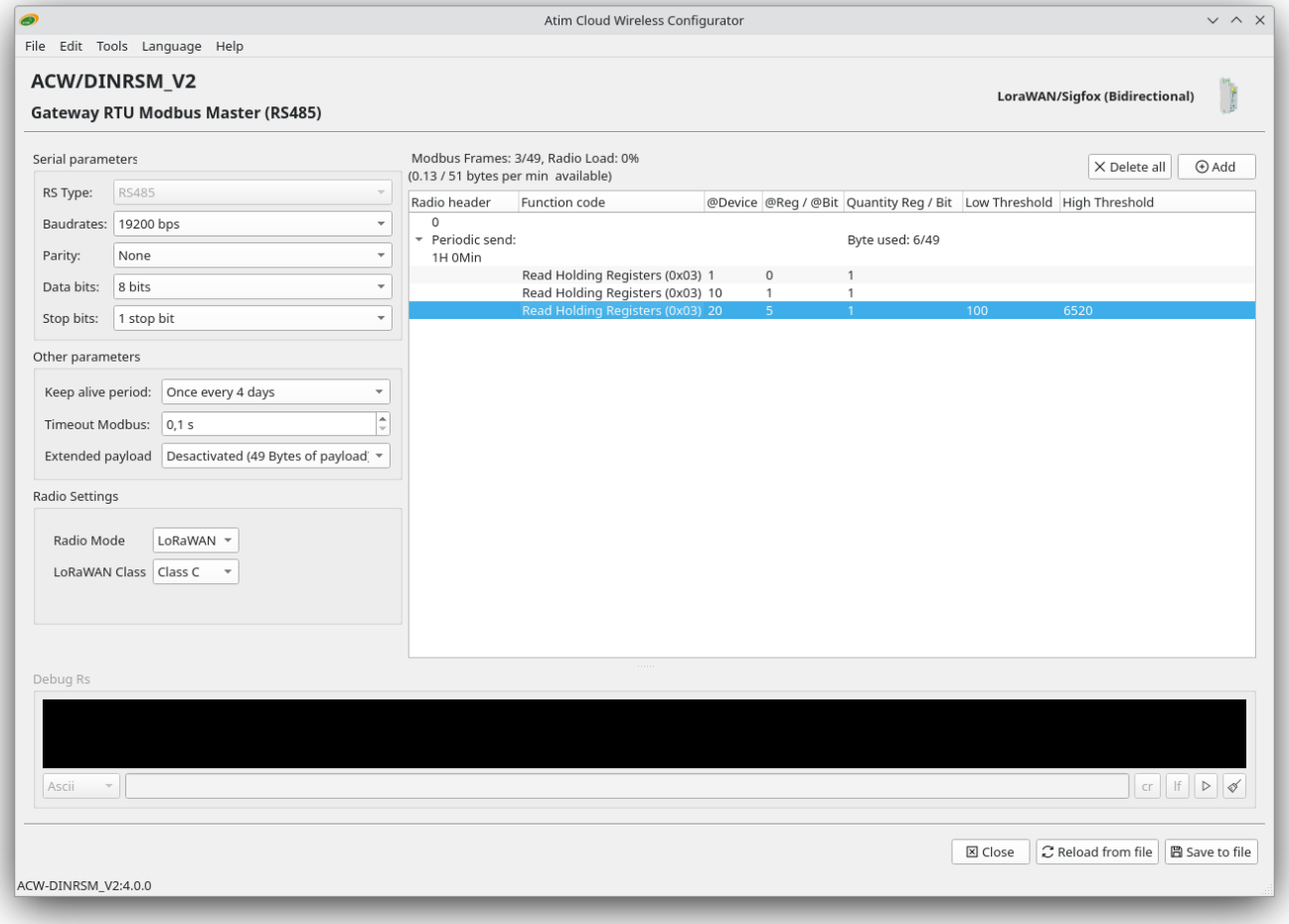
If we add a query in the same radio frame with:

- request to read register 0x04
- Modbus slave address 10
- Reading 5 registers from register 1



The configuration downlink for the second modbus request to be sent is:
D01100830A0100000A00010000000000000000

With an additional query in the radio frame (with a high and low threshold configured):



The configuration downlink of the 3rd modbus request is:

D11100831405000001000100000000364007819

NOTE

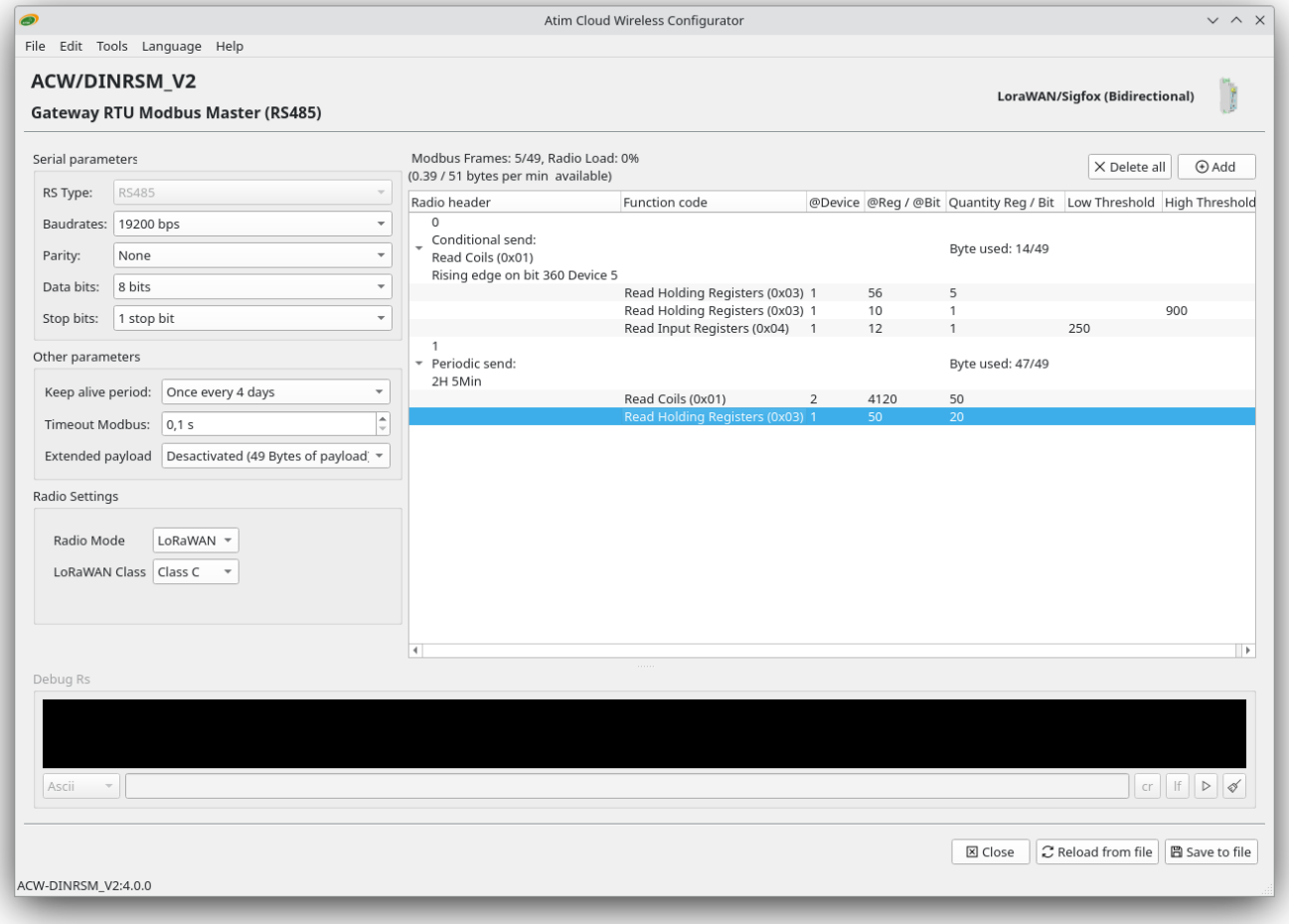
It is possible to concatenate downlinks into a single downlink, up to a limit of 51 bytes. In this example, if we want to minimize the number of downlinks to be sent in the case of a configuration of these 3 requests, we can concatenate them into 2 downlinks:

Downlink 1:
CF1100030100000001000100000BA0000000000D01100830A0100000100010000 6E0000000000D10B00831405000001000100000

Downlink 2:
D1070A6E0364007819

Request 3 is split into two (one part in the 1st downlink and one part in the second downlink). This is possible because all "modbus request" parameters are buffered parameters.

4.1.5.3.2 Cases with multiple radio frames and multiple queries per frame



In this configuration, there are 5 modbus requests spread over 2 radio frames.

Downlink query parameter 1: **CF11000301380000050105056801000000000Downlink query parameter 2: D0110083010A0000010105056801020000 8403Downlink query parameter 3: D1110084010C0000010105056801 01FA000000Downlink query parameter 4: D2110001021810013200020500650000000000Downlink query parameter 5: D3110083013200011400020500650000000000**

NOTE

It is possible to concatenate downlinks into a single downlink, up to a limit of 51 bytes. In this example, if we want to minimize the number of downlinks to be sent in the case of a configuration of these 5 requests, we can concatenate them into 2 downlinks:

Downlink 1:
CF11000301380000050105056801000000000D0110083010A0000010105056801020000840 3D10B0084010C00000101050568

Downlink 2:
D1070A0101FA000000D2110001021810013200020500650000000000D3110083013200011400020500650000000000

Request 3 is split into two (one part in the 1st downlink and one part in the second downlink). This is possible because all "modbus request" parameters are buffered parameters.

5 Control frame

5.1 List of product-specific orders

Order	Value (hexa)	waist
Modbus Controls	0x7F	Minimum 5

5.1.1 0x7F Modbus control

Byte 0 - Header	Byte 1 - Size	Byte 2 - order type	Bytes
0xC1 (0x01 order | 0xC0)	Waist	0x7F (Modbus Command)	Request

With "**Size**" the frame size from byte 2.

With "**Query**" as described:

Byte 0	Byte 1	Byte 2 - 3	Byte 4	Bytes
Address	Code	Register	Size	Value

With the "**Address**" field, the address of the modbus slave.

Value	Code
0x01	Modbus Function Code 0x01 (Read Coils)
0x02	Modbus Function Code 0x02 (Read Discrete Inputs)
0x03	Modbus Function Code 0x03 (Read Holding Registers)
0x04	Modbus Function Code 0x04 (Read Input Registers)
0x0F	Modbus code function 0x0F (Write Coils)
0x10	Modbus Function Code 0x10 (Write Holding Registers)

The "**Register**" field corresponds:

- to the address of the 1st register to be read/written (in the case of 0x03, 0x04, 0x10 codes)
- To the address of the 1st bit to be read/write (in the case of 0x01, 0x02, 0x0F codes)

The "**Size**" field corresponds to the number of registers or bits to read/write.

The "**Value**" field is only to be filled in for function codes 0x0F, 0x10 to specify the value of registers/bits to be written. For other requests, these bytes must not be present in the command to be sent.

Example

Request 9-bit Read on slave 1 from bit 211:

- Downlink to send: **0xC1067F0101D30009**

- Answer: **0x077FXXXX** with **XXXX** = 0bX217X216X215X214X213X213X212X211 0b0000000X218

Request to Read 2 registers on slave 1 from register 15:

- Downlink to send: **0xC1067F01030F0002**

- Answer: **0x077FXXXXYYYY** with **XXXX** register value 15 and **YYYY** register value 16

Request to write 15 bits to slave 2 from bit 520:

- Downlink to send: **0xC1087F020F08020FXXXX** with **XXXX** = 0bX527X526X525X524X523X522X521X520 0b0X534X533X532X531X530X529X528

- Answer: **0x077F0F**

Request to write 2 registers to slave 2 from bit 456 (0x0006 for register 456 and 0x1A98 for register 457):

- Downlink to send: **0xC10A7F0210C8010200061A98**