Commands RFU6xx

Version history

Version	Date	Author	Description
V1.00	20.01.2015	A.Pfeffer	Initial version
T2.00RC01	12.01.2016	A.Pfeffer	Additional information ▪ Command language ▪ Memory mapping ▪ Parameter specification ▪ Return values

Content

1. Command Language

The RFU6xx communicates with the host system via commands, which are defined by the command language **CoLa-A** (**Co**mmand **La**nguage **A**scii). Sometimes the commands are also called "*SOPAS-commands*", because the communication with the engineering tool SOPAS-ET is based on the same commands. This chapter describes the most important aspects of CoLa-A.

1.1. General structure of SOPAS telegram

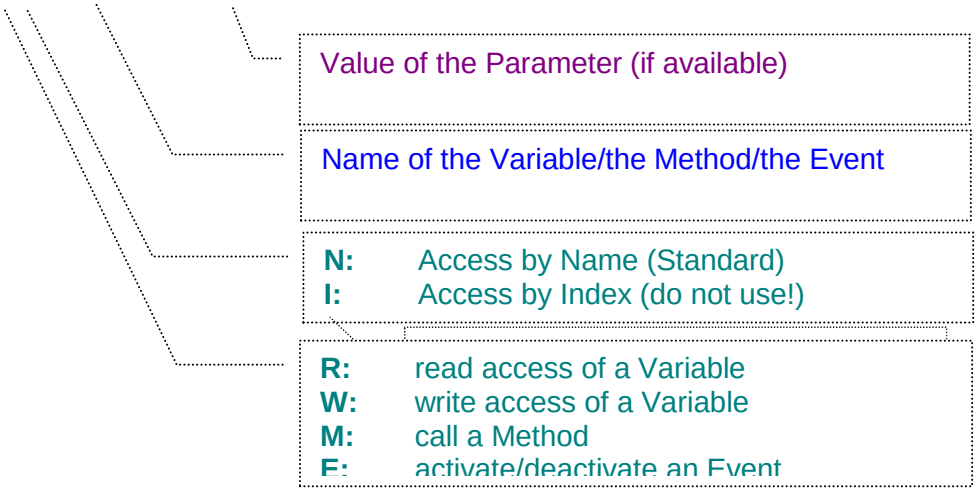
Each SOPAS telegram includes

- the TYP of the request (e.g. sMN),
- the Name of the Methode/Variable and
- the Parametern.

The type indicates whether the request is a variable, a method or an event.
The first character is always the lower case „s".
SOPAS-Commands are always framed with STX/ETX.

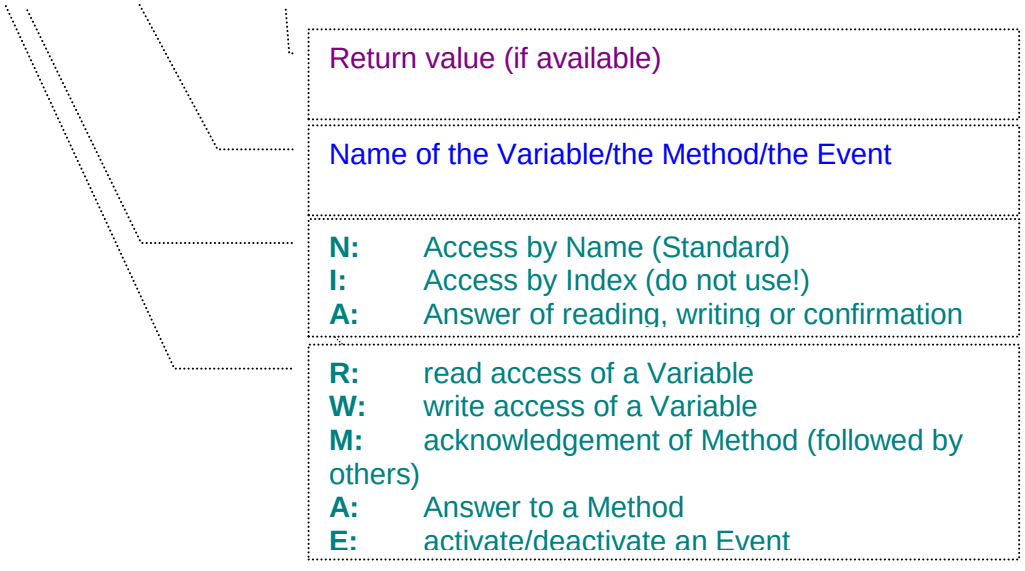
SOPAS-Telegram (for sending)

[STX]sWN FTScanFrq 10[ETX]



SOPAS-Telegram (for receiving)

[STX]sAN SetAccessMode 1[ETX]



1.2. Error codes

If an error occurs the Sopas command is answered with:
[STX]sFA <x>[ETX] as an error response

Error list: **<x>** can have following values:

1:	Access denied
2:	Unknown Index
3:	Unknown Index
4:	Wrong Condition
5:	Invalid Data
6:	Unknown Error
7:	Too Many Parameter
8:	Parameter Missing
9:	Wrong Parameter
A:	No Write Access
B:	Unknown Command
C:	Unknown Command
D:	Server Busy
E:	Textstring Too Long
F:	Unknown Event
10:	Too many Parameter
11:	Invalid Character
12.	No Message
13:	No Answer
14:	Internal Error
15:	HubAddress: wrong
16:	HubAddress: error
17:	HubAddress: error

1.3. Special command language issues

Numbers in command parameters are generally interpreted as hexadecimal values. If you want to send a parameters in decimal, you need to add a “+” symbol in front of it.

Example:
These two commands are equal.

```
sMN TAreadTagData 0 3 0 20 32
sMN TAreadTagData 0 3 0 +32 32
```

Numbers in command answers are always send as hexadecimal values.

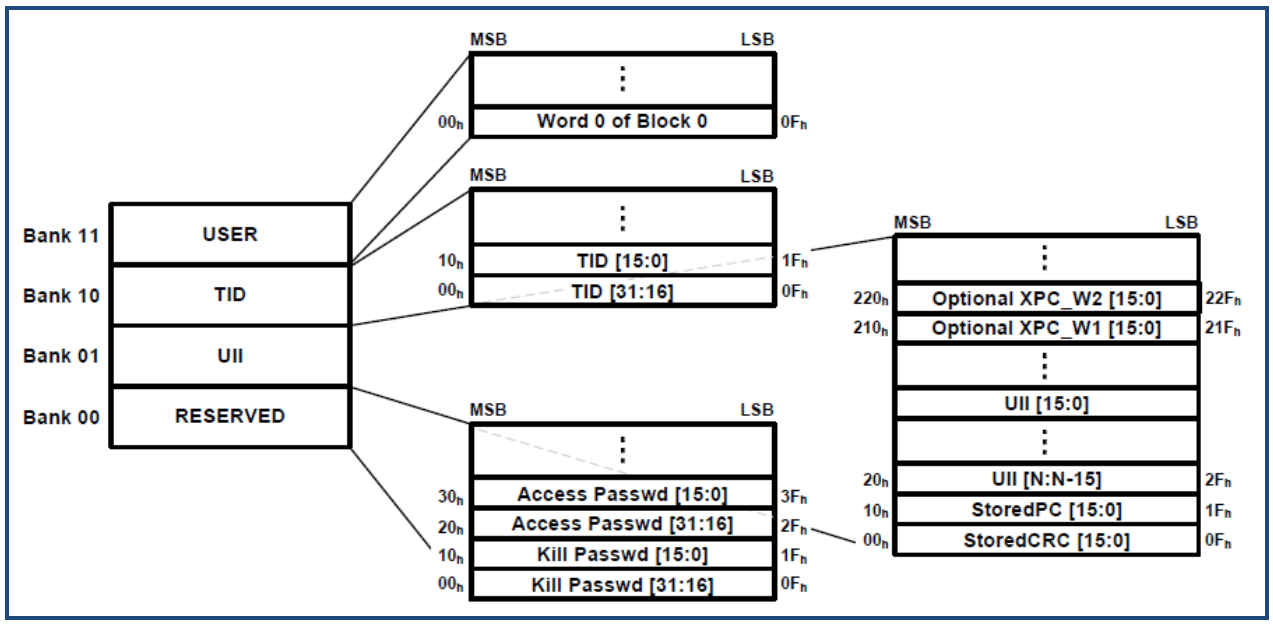
Example:

```
sAN TAreadTagData 1 18 AAAABBBBCCCCDDDDEEEEFFFF
```

2. Transponder memory mapping

2.1 General structure

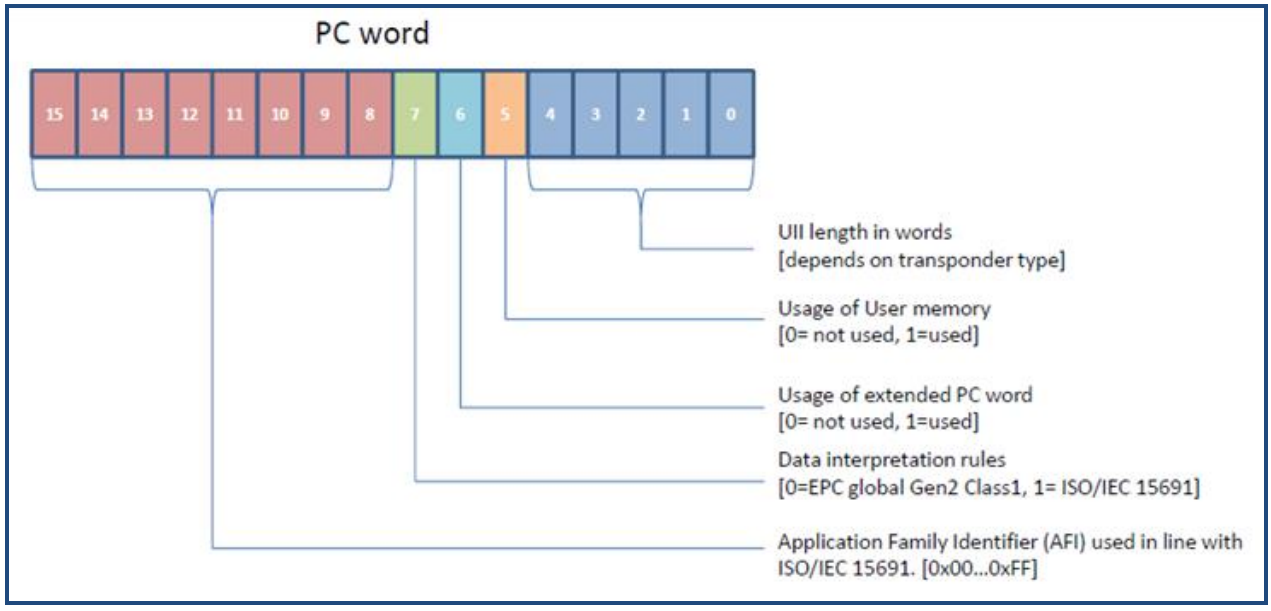
The following figure shows the memory structure of an UHF transponder (ISO/IEC 18000-6C).



The transponder memory is scales in words (1 word = 2 byte = 16 bit).
Please be aware that the UII is not equal to the UII memory bank, it is only part of it. To address the UII, you need to address the EPC/UII-memory bank with an offset of “2”.

2.2 PC-word

The exact meaning of the PC-word is shown in the figure below. It is positioned in the EPC/UII memory bank with an offset of “1”.



3. Commands RFU6xx

3.1 TAreadTagData

This is the basic command to read transponder data. For addressing the memory 16-bit words are used. The operation can be executed in addressed and non addressed mode. Addressed mode is recommended for the most of applications.
Attention: Non addressed mode must only be used, if it is impossible that there is more than on transponder in the field. Also if only one tag is supposed to be read, but it is possible to read another one because of overreaches, the command should be sent in addressed mode.

Parameter	Data type	Description
selector	Flex string	Specifies specific tag for the operation according to EPC bank starting at word 1 (PC-word). To send a non addressed command, the selector is set to zero (That means the length of selector is 0 and selector is empty)
bank	Enum8	Memory bank (0=passwords, 1=EPC/UII, 2=TID, 3=user memory)
pointer	UDInt	First word
wordcount	USInt	Number of words
retry	USInt	Number or retries, until failure is reported ▪ LoNibble: retries on the same frequency channel [0…7] ▪ HiNibble: frequency channel hoppings [0…5] Usually the default retry 0x32 is sufficient for most applications.

Return value	Data type	Description
valid	bool	Indicates if command was successful or not (0=NOK, 1=OK)
data	Flex string	Requested transponder memory data

Example 1: Read UII non addressed
To send a non addressed command, the selector is set to `0` (length of selector is 0 and selector is empty)

```
sMN TAreadTagData 0 1 2 6 32
```

`0` non addressed mode
`1` memory bank 1 (UII/EPC)
`2` offset 2 words (start of UII/EPC, because first two words are CRC and PC word)
`6` number of words to read (6 words)
`32` retries (3 frequency channel hoppings with 2 retries on each channel)

```
sAN TAreadTagData 1 18 111122223333444455556666
```

`1` command successful
`18` length of data in hex (0x18 = 24 charactetrs)
`111122223333444455556666` data

Example 2: Read user memory addressed

```
sMN TAreadTagData +28 3000111122223333444455556666 3 0 20 32
```

`+28` length of UII/EPC with PC word in characters
`3000111122223333444455556666` PC word + UII/EPC for addressing
`3` memory bank 3 (user memory)
`0` no offset, beginning of the memory bank
`20` number of words to read (32 words = 0x20)
`32` retries (3 frequency channel hoppings with 2 retries on each channel)

```
sAN TAreadTagData 1 18 AAAABBBBCCCCDDDDEEEEFFFF
```

`1` command successful
`18` length of data in hex (0x18 = 24 charactetrs)
`AAAABBBBCCCCDDDDEEEEFFFF` data

3.2 TAextReadTagData

This is an extended command to read transponder data. Additional to the basic functionality an offset for addressing can be set. So the transponder can be addressed by its UII/EPC without using the PC word. Further the antenna which is used for the operation can be specified.

Parameter	Data type	Description
offset	USInt	Defines the offset for the selector (e.g. 1 includes PC bits, 2 starts with UII)
selector	Flex string	Specifies specific tag for the operation according to EPC bank starting at word 1 (PC-word). To send a non addressed command, the selector is set to zero (That means the length of selector is 0 and selector is empty)
bank	Enum8	Memory bank (0=passwords, 1=EPC/UII, 2=TID, 3=user memory)
pointer	UDInt	First word
wordcount	USInt	Number of words
retry	USInt	Number or retries, until failure is reported ▪ LoNibble: retries on the same frequency channel [0…7] ▪ HiNibble: frequency channel hoppings [0…5] Usually the default retry 0x32 is sufficient for most applications.
antenna mask	USInt	Specifies the antenna for the requested action. [1=ant.1 , 2 = ant.2 , 3=ant.1+2, 4=ant.3 , …, 8=ant.4, …, F=ant. 1+2+3+4]

Return value	Data type	Description
valid	bool	Indicates if command was successful or not (0=NOK, 1=OK)
data	Flex string	Requested transponder memory data

Example 1: Read UMEM addressed with 6 words UII/EPC without using PC word

```
sMN TAextReadTagData 2 +24 111122223333444455556666 3 0 10 32 1
```

2 offset used for addressing (start of UII/EPC)
+24 length of UII/EPC in characters
111122223333444455556666 UII/EPC for addressing
3 memory bank 3 (user memory)
0 no offset, beginning of the memory bank
10 number of words to read (16 words = 0x10)
32 retries (3 frequency channel hoppings with 2 retries on each channel)
1 antenna 1

```
sAN TAextReadTagData 1 18 1111222233334444555566667777888899990000
```

1 command successful
18 length of data in hex (0x18 = 24 charactetrs)
1111222233334444555566667777888899990000 data

3.3 TAwriteTagData

This is the basic command to write data on a transponder. For addressing the memory 16-bit words are used. The operation can be executed in addressed and non addressed mode. Addressed mode is recommended for the most applications.
Attention: Non addressed mode must only be used, if it is impossible that there is more than on single transponder in the field.

Parameter	Data type	Description
selector	Flex string	Specifies specific tag for the operation according to EPC bank starting at word 1 (PC-word). To send a non addressed command, the selector is set to zero (That means the length of selector is 0 and selector is empty)
bank	Enum8	Memory bank (0=passwords, 1=EPC/UII, 2=TID, 3=user memory)
pointer	UDInt	First word
wordcount	USInt	Number of words
retry	USInt	Number or retries, until failure is reported

		▪ LoNibble: retries on the same frequency channel [0…7] ▪ HiNibble: frequency channel hoppings [0…5] Usually the default retry 0x32 is sufficient for most applications.
data	Flex string	Data to write

Return value	Data type	Description
valid	bool	Indicates if command was successful or not (0=NOK, 1=OK)
wordcount	USInt	Number of words written to the transponder successfully

Example 1: Write UMEM non addressed

```
sMN TAwriteTagData 0 3 0 5 32 +20 11112222333344445555
```

0 non addressed mode
3 memory bank 3 (user memory)
0 no offset, beginning of the memory bank
5 number of words to write (1 word)
32 retries (3 frequency channel hoppings with 2 retries on each channel)
+20 number of characters to write
11112222333344445555 data to write

```
sAN TAwriteTagData 1 5
```

1 command successful
5 5 words written successfully

Example 2: Write UMEM addressed with 6 words UII/EPC

```
sMN TAwriteTagData +28 3000111122223333444455556666 3 0 5 32 +20 11112222333344445555
```

+28 length of UII/EPC with PC word in characters
3000111122223333444455556666 PC word + UII/EPC for addressing
3 memory bank 3 (user memory)
0 no offset, beginning of the memory bank
5 number of words to write (5 words)
32 retries (3 frequency channel hoppings with 2 retries on each channel)
+20 number of characters to write
11112222333344445555 data to write

```
sAN TAwriteTagData 1 5
```

1 command successful
5 5 words written successfully

3.4 TAextWriteTagData

This is an extended command to write data on the transponder. Additional to the basic functionality an offset for addressing can be set. So the transponder can be addressed by its UII/EPC without using the PC word. Further the antenna which is used for the operation can be specified.

Parameter	Data type	Description
offset	USInt	Defines the offset for the selector (e.g. 1 includes PC bits, 2 starts with UII)
selector	Flex string	Specifies specific tag for the operation according to EPC bank starting at word 1 (PC-word). To send a non addressed command, the selector is set to zero (That means the length of selector is 0 and selector is empty)
bank	Enum8	Memory bank (0=passwords, 1=EPC/UII, 2=TID, 3=user memory)
pointer	UDInt	First word
wordcount	USInt	Number of words
retry	USInt	Number or retries, until failure is reported ▪ LoNibble: retries on the same frequency channel [0…7] ▪ HiNibble: frequency channel hoppings [0...5]

		Usually the default retry 0x32 is sufficient for most applications.
data	Flex string	Data to write
antenna mask	USInt	Specifies the antenna for the requested action. [1=ant.1 , 2 = ant.2 , 3=ant.1+2, 4=ant.3 , …, 8=ant.4, …, F=ant. 1+2+3+4]

Return value	Data type	Description
valid	bool	Indicates if command was successful or not (0=NOK, 1=OK)
wordcount	USInt	Number of words written to the transponder successfully

Example 1: Write UMEM addressed with 6 words UII/EPC without using PC word

```
sMN TAextWriteTagData 2 +24 111122223333444455556666 3 0 5 32 +20 1111222233344445555 1
```

2 offset used for addressing (start of UII/EPC)
+24 length of UII/EPC in characters
111122223333444455556666 UII/EPC for addressing
3 memory bank 3 (user memory)
0 no offset, beginning of the memory bank
5 number of words to write (5 words)
32 retries (3 frequency channel hoppings with 2 retries on each channel)
+20 number of characters to write
1111222233344445555 data to write
1 antenna 1

```
sAN TAextWriteTagData 1 5
```

1 command successful
5 5 words written successfully

3.5 TAadvWriteTagData

This is an advanced write command to write data on the transponder. Additional to the functionality of the *TAextWriteTagData* command, it is possible to address the transponder by any memory bank.
After a write operation on the UII/EPC memory bank fails, the content of the UII/EPC is undefined. A retry won't be successful here, because the previous UII/EPC is still used. With this command the transponder can be addressed on its TID memory bank. So the write operation can be repeated in case of a write error. It is recommended to use this command for all write operations on the UII/EPC memory bank.

Parameter	Data type	Description
Selector_bank	Enum8	Memory bank of the selector (1=EPC/UII, 2=TID)
offset	USInt	Defines the offset for the selector (e.g. 1 includes PC bits, 2 starts with UII)
selector	Flex string	Specifies specific tag for the operation according to EPC bank starting at word 1 (PC-word). To send a non addressed command, the selector is set to zero (That means the length of selector is 0 and selector is empty)
bank	Enum8	Memory bank (0=passwords, 1=EPC/UII, 2=TID, 3=user memory)
pointer	UDInt	First word
wordcount	USInt	Number of words
retry	USInt	Number or retries, until failure is reported ▪ LoNibble: retries on the same frequency channel [0…7] ▪ HiNibble: frequency channel hoppings [0…5] Usually the default retry 0x32 is sufficient for most applications.
data	Flex string	Data to write
antenna mask	USInt	Specifies the antenna for the requested action. [1=ant.1 , 2 = ant.2 , 3=ant.1+2, 4=ant.3 , …, 8=ant.4, …, F=ant. 1+2+3+4]

Return value	Data type	Description
valid	bool	Indicates if command was successful or not (0=NOK, 1=OK)
wordcount	USInt	Number of words written to the transponder successfully

Example 1: Write UII addressed with 4 words TID

```
sMN TAadvWriteTagData 2 0 +16 E20068060DF667AE 1 1 7 32 +28 3000111122223333444455556666 1
```

2 memory bank 2 for addressing (TID)
0 offset used for addressing (start of TID memory bank)
+16 length of TID in characters
E20068060DF667AE TID for addressing
1 memory bank 1 (UII/EPC)
1 offset 1 word (PC word)
7 number of words to write (7 words, PC word + 6 words of UII)
32 retries (3 frequency channel hoppings with 2 retries on each channel)
+28 number of characters to write
3000111122223333444455556666 data to write
1 antenna 1

```
sAN TAadvWriteTagData 1 7
```

1 command successful
7 7 words written successfully

3.6 TAusePasswd

To protect a transponder against unauthorized access, the transponder supports a password management.
If the memory banks of the transponder should be locked, a *access password* has to be defined first.
To kill a transponder a *kill password* has to be defined first.
Each password has 8 digits. If the password is “00000000”, that means that no password is used.

There are passwords in the transponder and there are passwords in the device too. They have to match. With the following command, the Device Passwords in the RFU6xx can be defined.

Example 1: Set Device Access Password to “12345678”

```
sMN TAusePasswd 1 12345678
```

1 Define transponder access password
12345678 Password data

Example 2: Set Device Kill Password to “12345678”

```
sMN TAusePasswd 0 12345678
```

0 Define transponder kill password
12345678 Password data

3.7 TAlockTagData

It is possible to lock the UII/EPC and the USER memory bank, such as the transponder passwords.
All memory banks can be locked temporarily or permanently. Only if the memory is locked temporarily, it can be unlocked again. The command contains a parameter *data* which defines the locking action. The following table helps to set this parameter:

memory bank	**action**	**temporary/permanent**	**data parameter**
UII	lock	temporary	4 8020
UII	lock	permanent	4 C030
USER	lock	temporary	3 802
USER	lock	permanent	3 C03
UII + USER	lock	temporary	4 8822
UII + USER	lock	permanently	4 CC33
Tag access password	lock	temporary	5 20080
Tag access password	lock	permanent	5 300C0
Tag kill password	lock	temporary	5 80200
Tag kill password	lock	permanent	5 C0300
All memory banks	lock	temporary	5 A8AA2

All memory banks	lock	permanent	5 FCFF3
UII	unlock	temporary	4 8000
UII	unlock	permanent	4 C010
USER	unlock	temporary	3 800
USER	unlock	permanent	3 C01
UII + USER	unlock	temporary	4 8800
UII + USER	unlock	permanently	4 CC11
Tag access password	unlock	temporary	5 20000
Tag access password	unlock	permanent	5 30040
Tag kill password	unlock	temporary	5 80000
Tag kill password	unlock	permanent	5 C0100
All memory banks	unlock	temporary	5 A8800
All memory banks	unlock	permanent	5 FCD51

Example 1: Lock UII/EPC memory bank permanently

```
sMN TAlockTagData +28 3000111122223333444455556666 32 4 C030
```

`+28` Length of UII/EPC with PC word in characters

`3000111122223333444455556666` PC word + UII/EPC for addressing

`32` Retries (3 frequency channel hoppings with 2 retries on each channel)

`4` Length of data

`C030` Data (see table above), here lock UII/EPC memory bank permanently

3.8 TAextLockTagData

This is an extended command to lock transponder data. Additional to the basic functionality an offset for addressing can be set. So the transponder can be addressed by its UII/EPC without using the PC word. Further the antenna which is used for the operation can be specified. The rest is equal to the **`TAlockTagData`** command. For further details see description above.

Example 1: Lock UII/EPC memory bank permanently

```
sMN TAextLockTagData 2 +24 111122223333444455556666 32 4 C030 1
```

`2` Offset used for addressing (start of UII/EPC)

`+24` Length of UII/EPC with PC word in characters

`111122223333444455556666` PC word + UII/EPC for addressing

`32` Retries (3 frequency channel hoppings with 2 retries on each channel)

`4` Length of data

`C030` Data (see table above), here lock UII/EPC memory bank permanently

`1` Antenna 1 (1=Antenna1 , 2 = Antenna2 , 4=Antenna3 , 8=Antenna4)

3.9 TAkillTagData

This command destroys the transponder irreversible.
Attention: The transponder will not be accessible any more after executing this command.

Example 1: Kill transponder

```
sMN TAkillTagData 2 +24 111122223333444455556666 32 1
```

`2` Offset used for addressing (start of UII/EPC)

`+24` Length of UII/EPC in characters

`111122223333444455556666` UII/EPC for addressing

`32` Retries (3 frequency channel hoppings with 2 retries on each channel)

`1` Antenna 1 (1=Antenna1 , 2 = Antenna2 , 4=Antenna3 , 8=Antenna4)

3.10 IVSingleInv

This command performs a single inventory on all defined antennas.

Example 1: Inventory on internal antenna only

```
sMN IVSingleInv 1
```

```
1               Selected antennas for inventory (1=Ant1 , 2=Ant2 , 3=Ant1+Ant2, …, F=all)

sAN IVSingleInv 1 1 1C 300011112222333344445555666 1 FE4C 0 0 0 96 0 0 0
1               Inventory was successful
1               Number of detected transponders
1C              Length of UII/EPC with PC-word
300011112222333344445555666   UII/EPC in HEX with PC-word
1               Antenna ID (1=Antenna1 , 2 = Antenna2 , 4=Antenna3 , 8=Antenna4)
FE4C 0 0 0      RSSI value for each antenna (0xFFFF-0xFE4C=0x1B3 → 435=-43,5=dBm)
96 0 0 0        antenna power for each antenna scaled in 10dBm (0x96=150 → 15dBm)
```

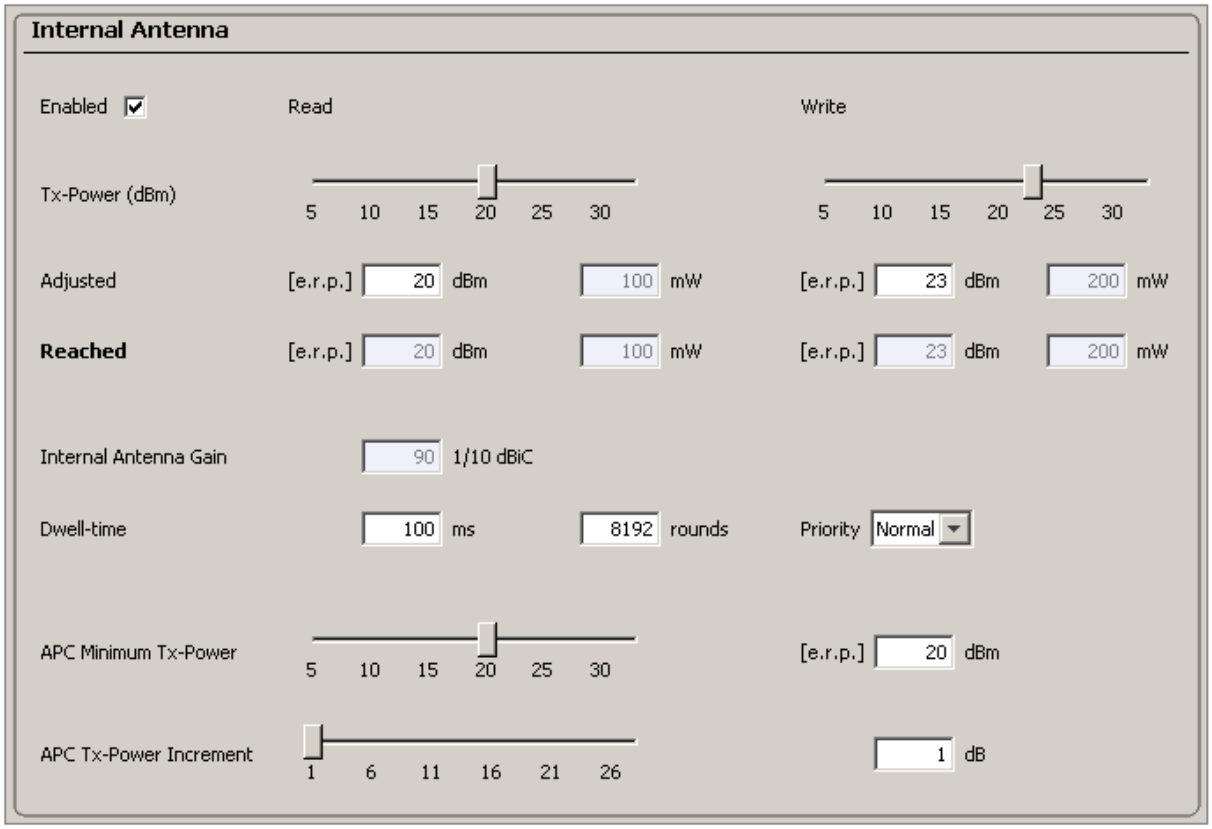
Example 2: Inventory on all antennas

```
sMN IVSingleInv F
F               Selected antennas for inventory (1=Ant1 , 2=Ant2 , 3=Ant1+Ant2, …, F=all)

sAN IVSingleInv 1 2 1C 300011112222333344445555666 1 FE4C 0 0 0 96 0 0 0 1C
31A1AAAABBBBAAAABBBBAAAA0001 2 0 FE04 0 0 0 64 0 0

1               Inventory was successful
2               Number of detected transponders
1C              Length of UII/EPC with PC-word
300011112222333344445555666   UII/EPC in HEX with PC-word
1               Antenna ID (1=Antenna1 , 2 = Antenna2 , 4=Antenna3 , 8=Antenna4)
FE4C 0 0 0      RSSI value for each antenna (0xFFFF-0xFE4C=0x1B3 → 435=-43,5=dBm)
96 0 0 0        antenna power for each antenna scaled in 10dBm (0x96=150 → 15dBm)
1C              Length of UII/EPC with PC-word
31A1AAAABBBBAAAABBBBAAAA0001   UII/EPC in HEX with PC-word
2               Antenna ID (1=Antenna1 , 2 = Antenna2 , 4=Antenna3 , 8=Antenna4)
0 FE04 0 0      RSSI value for each antenna (0xFFFF-0xFE04=0x1FB → 507=-50,7=dBm)
0  64 0 0       antenna power for each antenna scaled in 10dBm (0x64=100 → 10dBm)
```

3.11 ADconfig0 (RFU620, RFU630-13xxx) / ADconfig1…ADconfig4 (RFU630-04xxx)

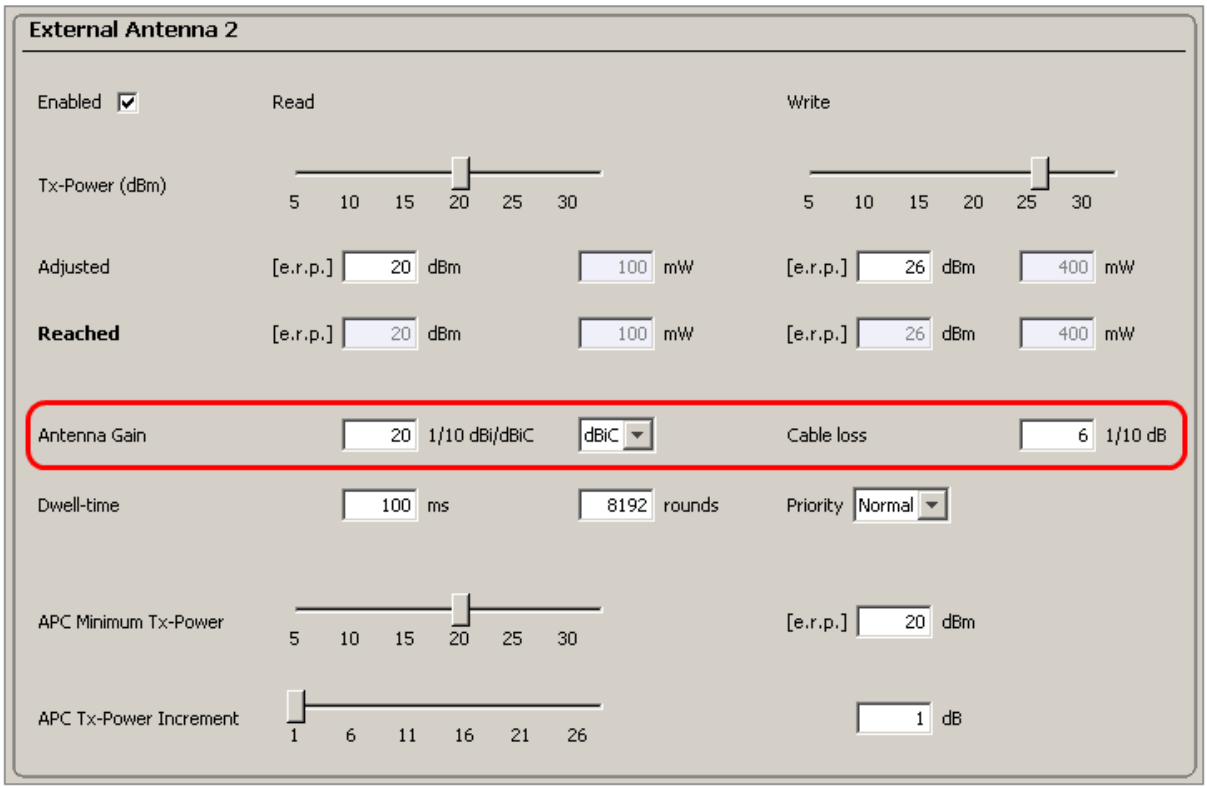
These variables are to read and change the antenna settings. To write on the variable a login as authorised client or higher is required.

Example 1: Set internal antenna (ADconfig0)




```
sWN ADconfig0 1 64 C8 E6 2000 1 C8 A 0
1      Antenna enabled
64     Dwell time 100ms (0x64=100)
C8     Read power in 10dBm (0xC8=200 → 20dBm)
E6     Write power in 10dBm (0xE6=230 → 23dBm)
2000   8192 Inventory rounds (0x2000=8192)
1      Antenna priority normal (0=low, 1=normal, 2=high)
C8     Minimum power for APC
A      Power increment for APC
0      Reserved

sMN SetAccessMode 3 F4724744
sAN SetAccessMode 1
sWN ADconfig0 1 64 C8 E6 2000 1 C8 A 0
sWA ADconfig0
sMN Run
sAN Run 1
```

Example 2: Set external antenna (ADconfig1…ADconfig4)

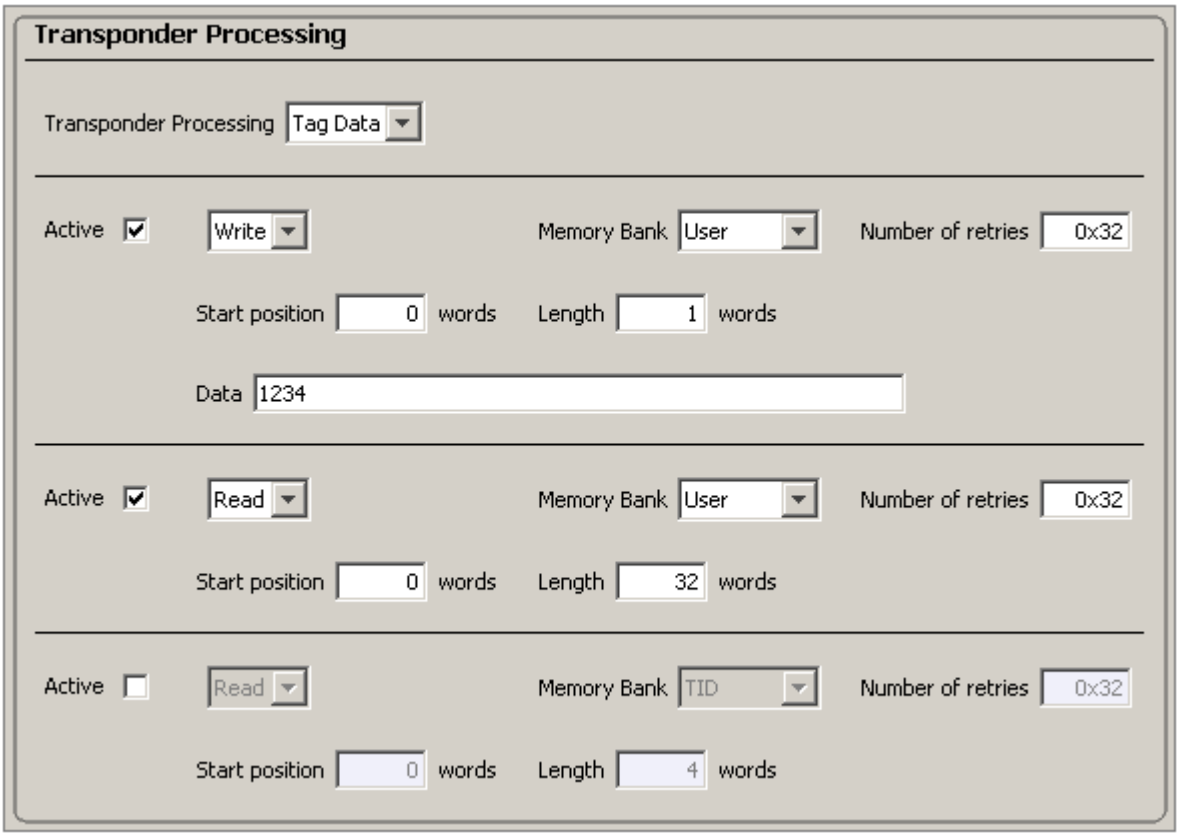



```
sWN ADconfig2 1 64 C8 104 6 14 0 2000 1 C8 A 0
1      Antenna enabled
64     Dwell time 100ms (0x64=100)
C8     Read power in 10dBm (0xC8=200 → 20dBm)
104    Write power in 10dBm (0x104=260 → 26dBm)
6      Cable loss in 1/10dB (0x06=6 → 0,6dB)
14     Antenna gain in 1/10dBi/dBiC (0x14=20 → 2dBi/dBiC)
0      Antenna gain unit (0=dBiC, 1=dBi)
2000   8192 Inventory rounds (0x2000=8192)
1      Antenna priority normal (0=low, 1=normal, 2=high)
C8     Minimum power for APC
A      Power increment for APC
0      Reserved

sMN SetAccessMode 3 F4724744
sAN SetAccessMode 1
sWN ADconfig2 1 64 C8 104 6 14 0 2000 1 C8 A 0
sWA ADconfig2
sMN Run
sAN Run 1
```

3.12 TPProcessingConfig

When sending read/write commands in dynamic applications, the transponder has to be positioned in the field already. Otherwise the command will fail. For these requirements the RFU6xx offers an alternative way to read/write transponder data using Transponder Processing. Here the read/write operation will be started automatically after a transponder has been detected. So the reading gate can also be started, before the transponder moves in the field. To parameterize the Transponder Processing, SOPAS-ET can be used (see screenshot “Example 1” below). If always the same action has to be executed, e.g. reading the UMEM, the Transponder Processing can be parameterized once. For write operations usually the data have to be defined by the control unit and they change with each reading gate. In this case the parameterization has to be done using the command “TPProcessingConfig” before each reading gate. Therefore a login is necessary. Within the next reading gate the Transponder Processing is executed automatically and the data are written into the transponder.

Example 1: Set Transponder Processing



```
sWN TPProcessingConfig 1 1 3 0 1 32 4 1234 1 0 3 0 20 32 0 0 0 2 0 4 32 0
1     Action 1 active
1     Write operation
3     Memory bank 3 (UMEM)
0     Offset 0 words
1     Length 1 word
32    Retries (3 frequency channel hoppings with 2 retries on each channel)
4     Length of Data to write
1234  Data to write on transponder
1     Action 2 active
0     Read operation
3     Memory bank 3 (UMEM)
0     Offset 0 words
20    Length 32 words (0x20=32)
32    Retries (3 frequency channel hoppings with 2 retries on each channel)
0     Data to write (not necessary for read operation)
0     Action 3 not active
0     Read operation
2     Memory bank 2 (TID)
0     Offset 0 words
4     Length 32 words (0x04=4)
32    Retries (3 frequency channel hoppings with 2 retries on each channel)
0     Data to write (not necessary for read operation)

sMN SetAccessMode 3 F4724744
sAN SetAccessMode 1
sWN TPProcessingConfig 1 1 3 0 1 32 4 1234 1 0 3 0 20 32 0 1 0 2 0 4 32 0
sWA TPProcessingConfig
sMN Run
sAN Run 1
```