READER-HOST-PROTOCOL
PUR - EXTENSION

iDTRONIC GmbH · Donnersbergweg 1 · 67059 Ludwigshafen · support@idtronic.de · idtronic-group.com

History of Change:

10.03.2010	R&D	Created document	v0.01
21.06.2011	R&D	Added Lock-Tag codes	v0.02
08.12.2011	R&D	Added Gen2-EPC-Size, Added description for Read-From-Tag, Added notification values	v0.03
12.12.2012	R&D	Added Cyclic-Inventory Start Bytes Added Gen2-Send-Handle Added Gen2-Send-PC	v0.04
09.01.2013	R&D	Added Commands: • Get-Handle-From-Tag • Read-From-Handle • Write-To-Handle	v0.05
15.01.2013	R&D	Added new Inventory Mode Added new Tag-Handler	v0.06
02.05.2013	R&D	Added Gen2 Settings: • Gen2-Q-Setting • Gen2-Q-Method • Gen2-Session • Gen2-InventoryCount	v0.07
10.10.2013	R&D	Added Custom-Tag-Command	v0.08
26.06.2014	R&D	Added Gen2 Setting: Selection Mask	v0.09
28.07.2014	R&D	Added Gen2-BlockWrite and PostDetect-READ	v0.10
22.01.2015	R&D	Added some optional parameters	v0.11
19.03.2015	R&D	Added read block size parameter	v0.12

iDTRONIC GmbH · Donnersbergweg 1 · 67059 Ludwigshafen · support@idtronic.de · idtronic-group.com

1 Introduction 5

2 Parameter-Dictionary 6

2.1 Inventory Mode 7

2.2 Power Safe Setting 8

2.3 RSSI 8

2.4 Tag-Id-Behavior 9

2.5 Gen2-Link-Frequency 11

2.6 Gen2-Bit-Encoding 11

2.7 Gen2-Modulation Depth 12

2.8 Gen2-EPC-Size 12

2.9 Gen2-Send-Handle 13

2.10 Gen2-Send-PC 13

2.11 Gen2-Q-Setting 13

2.12 Gen2-Q-Method 14

2.13 Gen2-Session 14

2.14 Gen2-InventoryCount 14

3 Function Descriptions 17

3.1 Read-From-Tag (50-03) 17

3.2 Lock-Tag (50-05) 17

4 Custom-Tag-Commands 18

4.1 NXP-Set-ReadProtect (01) 19

4.2 NXP-Clear-ReadProtect (02) 20

4.3 Get-Handle-From-Tag (04) 22

4.4 Read-From-Handle (05) 23

4.5 Write-To-Handle (06) 24

4.6 Custom-tag-Command-Handle (07) 25

5 Cyclic-Inventory Start Bytes 27

6 Notifications 28

iDTRONIC GmbH · Donnersbergweg 1 · 67059 Ludwigshafen · support@idtronic.de · idtronic-group.com

1 Introduction

This document describes the extensions to the standard UHF-RFID Reader-Host-Protocol for the PUR.

2 Parameter-Dictionary

The Parameters that can be set and read with the commands Get-Param and Set-Param are shown in the following table:

Name	Address	Size	Description	Version
Inventory Mode	0x0000	1 Byte	The used inventory mode	
Power Safe Settings	0x0001	3 Bytes	The settings for the power safe	
RSSI	0x0002	1 Byte	Defines if the RSSI Value is sent to the host	
Tag-Id-Behavior-Mode	0x0003	1 Byte	Defines how the reader should behave if a tag is detected.	
Gen2-Link-Frequency	0x0020	1 Byte	The used link frequency	
Gen2-Bit-Encoding	0x0021	1 Byte	The used bit encoding	
Gen2-Modulation-Depth	0x0022	1 Byte	The used modulation depth	
Gen2-EPC-Size	0x0023	1 Byte	The expected EPC size	V1.07
Gen2-Send-Handle	0x0024	1 Byte	Send Handle with EPC	V1.17
Gen2-Send-PC	0x0025	1 Byte	Send PC with EPC	V1.17
Gen2-Q-Setting	0x0026	3 Bytes	Q settings for the inventory	V2.01
Gen2-Q-Method	0x0027	1 Byte	Q adjust method	V2.01
Gen2-Session	0x0028	1 Byte	Gen2 session	V2.01
Gen2-InventoryCount	0x0029	1 Byte	Inventory Count	V2.01
Gen2-Select-Mask #1	0x002A	6 Bytes + Mask	Selection Mask #1 used for Gen2-Select command	V2.06
Gen2-Select-Mask #2	0x002B	6 Bytes + Mask	Selection Mask #2 used for Gen2-Select command	V2.06
Gen2-Read-Block-Size	0x002C	1 Byte	Count of words at Gen2 Read operation	V2.11

2.1 Inventory Mode

The set inventory mode defines how the reader does an inventory. This setting affects for example the used slot count and the used anti collision procedure.

Data structure:

1 byte Inventory-Mode-Enum

Possible settings:

Name	Value	Description
Gen2 – Fast Multi Tag	0x00	Inventory Mode that does not take the tag to the Opened but to the Acknowledged State. This inventory mode is not as secure as the standard mode, but is faster.
Gen2 – Fast Single Tag	0x01	The same inventory mode like the Fast Multi Tag, but with the slot count of 1. This has the effect that no anti-collision procedure is performed, but if there is only one tag in the field, it is detected much more faster
Gen2 – Standard Inventory	**0x02**	Inventory mode like defined in the standard
Gen2 – NXP-ReadProtect Inventory	0x03	Inventory that only searches for read protected tags. The EPC of these tags will always be composed of zeros.
Gen 2 – Standard Single Tag	0x04	The same inventory mode like the Standard Multi Tag, but with the slot count of 1. This has the effect that no anti-collision procedure is performed, but if there is only one tag in the field, it is detected much more faster.

Default:

Inventory-Mode-Enum **0x02** Gen2 – Standard Inventory

2.2 Power Safe Setting

The power safe setting can be used to pulse the reader and so safe power. If turned on, the reader switches the field on and performs an inventory after which he switches off the field and sleeps for the specified time. After this time this process is restarted.

Data Structure:

1 byte	Switch (ON (0x01) / OFF (0x00))
2 byte	Sleep Time in milliseconds

Default:

Switch	0x00	OFF
Sleep Time	0x00FA	250 ms

2.3 RSSI

If the RSSI Setting is enabled, the RSSI Value of the detected tags is appended to every Inventory-Cyclic-Interrupt.

Data Structure:

1 byte	Switch (ON (0x01) / OFF (0x00))

Default:

Switch	0x00	OFF

2.4 Tag-Id-Behavior

With this option the reader be configured how, it should react if it detects a tag.

Data structure:

1 byte Tag-Id-Behavior-Enum

Possible settings:

Name	Value	Description
Send-Tag-Id-Immediately	**0x00**	Every time the reader detects a tag, it is immediately forwarded to the host.
Send-Tag-Id-Once	0x01	If the reader detects a tag, it forwards the tag-Id to the host and stores it into a temporary buffer. If the same tag is detected again in the same inventory round, it is no more forwarded to the host. The buffer is cleared at the start of each cyclic inventory.
Send-Tag-Id-Immediately And Stop	0x02	The same Tag-Id behavior like Send-Tag-Id-Immediately but when the first tag is detected, the Cyclic-Inventory is stopped.

Default:

Tag-Id-Behavior-Enum **0x00** Send-Tag-Id-Immediately

2.5 Send Frequency

If this option is activated, the reader sends in a Cyclic-Inventory-Interrupt the frequency, where the tag was detected.

Data Structure:

1 byte Switch (ON (0x01) / OFF (0x00))

Default:

Switch 0x00 OFF

2.6 PostDetect-READ

The PostDetect-READ function offers the possibility to automatically read data from the detected tag directly after the tag was detected, so there are no more additional commands needed. The function reads the data every time a tag is detected. If the read was successful, the data is attached to the Cyclic-Inventory Interrupt.

Data Structure:

1 byte	Switch (ON (0x01) / OFF (0x00))
1 byte	Memory Bank
2 bytes	Address
1 byte	Byte-Count to Read
4 bytes	Password

Default:

Switch	0x00	OFF
Memory Bank	0x00	Reserved-Bank
Address	0x0000	Word 0
Size	0x00	0 Byte
Password	00-00-00-00	

Example:

To automatically read the data of the NXP G2iL status register, the following settings have to be set:

Switch	0x01	ON
Memory Bank	0x01	EPC-Bank
Address	0x0020	Word 32
Size	0x01	0 Byte
Password	00-00-00-00	

2.7 Gen2-Link-Frequency

This option sets the used Link Frequency for the Gen2 protocol.

Data Structure:

1 byte Link-Frequency

Possible settings:

Value	Description
0x00	40 kHz
0x01	80 kHz
0x02	160 kHz
0x03	213 kHz
0x04	256 kHz
0x05	320 kHz

Default:

Link-Frequency **0x02** 160 kHz

2.8 Gen2-Bit-Encoding

This option sets the used Bit-Encoding for the Gen2 protocol.

Data Structure:

1 byte Bit-Encoding

Possible settings:

Value	Description
0x00	FM0
0x01	Miller 2
0x02	Miller 4
0x03	Miller 8

Default:

Bit-Encoding **0x01** Miller 2

2.9 Gen2-Modulation Depth

This option sets the used Modulation Depth for the Gen2 protocol.

Data Structure:

1 byte Modulation-Depth in %

Default:

Modulation-Depth **0x64** 100 %

2.10 Gen2-EPC-Size

This option sets the expected EPC size of the tags that should be scanned. If the EPC size is set to a constant value, only tags with this EPC size are detected. If the EPC size is set to 0, all tags with different sizes are detected. For constant sizes only multiples of 2 are allowed.

Data Structure:

1 byte EPC Size

Possible settings:

Value	Description
0	Dynamic
2	2 Byte EPC
4	4 Byte EPC
…	…
0x01	12 Byte EPC
…	…
18	18 Byte EPC

Default:

Bit-Encoding **12** Constant 12 Bytes Size

2.11 Gen2-Send-Handle

If the Send-Handle is enabled, the Handle Value of a detected tag is appended to every Inventory-Cyclic-Interrupt.

Data Structure:

1 byte	Switch (ON (0x01) / OFF (0x00))

Default:

Switch	0x00	OFF

2.12 Gen2-Send-PC

If the Send-PC is enabled, the PC Value of a detected tag is appended to every Inventory-Cyclic-Interrupt.

Data Structure:

1 byte	Switch (ON (0x01) / OFF (0x00))

Default:

Switch	0x00	OFF

2.13 Gen2-Q-Setting

This field specifies the used Q value for Gen2 inventory. The Q defines how much slots are used for an inventory. The slot count is 2Q.

Data Structure:

1 byte	Initial Q
1 byte	Minimal Q
1 byte	Maximal Q

Default:

Initial Q	4	Slot count: 16
Minimal Q	2	Slot count: 4
Maximal Q	15	Slot count: 32768

2.14 Gen2-Q-Method

This field specifies the method which is used to adjust the Q value between different inventories.

Data Structure:

1 byte Method ID

Possible settings:

Value	Description
0x00	Static (Q stays at initial value)
0x01	RFE Dynamic Adjust Method

Default:

Method ID 0x01 RFE Dynamic Adjust Method

2.15 Gen2-Session

This field specifies the method used session for inventories.

Data Structure:

1 byte Session

Possible settings:

Value	Description
0x00	Session 0
0x01	Session 1
0x02	Session 2
0x03	Session 3

Default:

Session 0x01 Session 1

2.16 Gen2-InventoryCount

This field specifies how much inventories are executed until a new “Select” command is issued.

Data Structure:

1 byte Count

Default:

CountQ 0x01 1 inventory per Select cmd.

2.17 Gen2-Select-Mask #1

This field specifies the selection mask that should be used for the first Gen2-Select command. With the Gen2-Select command the tags in reach of the reader can be filtered in advance, so only the tags that apply to this mask will answer.

The Gen2-Select command is only sent before every “Multi Tag” inventory procedure.

Data Structure:

1 byte	**Memory Bank**
4 bytes	**Bit-Address**
1 byte	**Bit-Size of Mask**
N bytes	**Mask**

Default:

Memory Bank	0x01	EPC-Bank
Bit-Address	0x00	Bit 0
Bit-Size of Mask 0x00	0 Bit	
Mask	No mask	

Example:

The data for the filter that only tags with E2-7 in the first 12 Bits of the EPC should reply would be as follows:

Memory Bank	0x01	EPC-Bank
Bit-Address	0x20	Bit 32
Bit-Size of Mask	0x0C	12 Bits
Mask	0xE270	Mask E2-7

2.18 Gen2-Select-Mask #1

This field specifies the selection mask that should be used for the second Gen2-Select command. For more details see Gen2-Select-Mask #1

Data Structure:

1 byte	**Memory Bank**
4 bytes	**Bit-Address**
1 byte	**Bit-Size of Mask**
N byte	**Mask**

Default:

Memory Bank	0x01	EPC-Bank
Bit-Address	0x00	Bit 0
Bit-Size of Mask	0x00	0 Bit
Mask		No mask

2.19 Gen2-Read-Block-Size

The count of words that are read with one Gen2 Read operation.

Data Structure:

1 byte	**Count**

Default:

Count	0x01	1 word per Gen2 Read operation.

3 Function Descriptions

3.1 Read-From-Tag (50-03)

If the byte size 0 is selected, the reader reads until the tag responses with a “Memory Overrun” error. But with using tags with bigger User Memory banks, the reader uses another stop condition. It reads at a maximum of 200 bytes, and then returns the 200 bytes. So if bigger amounts of data should be read and the maximum size is not known, the Read-From-Tag function should be called with incremented addresses as long as the read byte count is lower than 200 bytes or a TMI_MEM_OVERRUN error is returned.

3.2 Lock-Tag (50-05)

The lock tag command needs to types of codes to lock a tag. For a Gen2 tag these codes are:

```
typedef enum{
      UNLOCK                  = 0x00,
      LOCK                    = 0x01,
      PERMALOCK               = 0x02,
      LOCK_AND_PERMALOCK      = 0x03,
} eRFE_LOCK_MODE;

typedef enum{
      KILL_PASSWORD     = 0x00,
      ACCESS_PASSWORD   = 0x01,
      EPC               = 0x02,
      TID               = 0x03,
      USER              = 0x04,
} eRFE_LOCK_MEMORY_SPACE;
```

These codes are directly connected to the Gen2 standard. Further information about these codes and there meaning can be found there.

4 Custom-Tag-Commands

In this section the available custom tag commands are documented. These commands can be used by calling the command 05-10 of the Reader-Host-Protocol.

The available commands are:

Command	Value
NXP-Set-ReadProtect	0x01
NXP-Clear-ReadProtect	0x02
Block-Write-To-Tag	0x03
Get-Handle-From-Tag	0x04
Read-From-Handle	0x05
Write-To-Handle	0x06
Custom-Tag-Command-Handle	0x07
Block-Write-To-Handle	0x08

4.1 NXP-Set-ReadProtect (01)

This function transfers a NXP tag into the ReadProtect Mode. In this mode the tag only returns zeroes instead of its actual EPC.

Parameters: unsigned char **command,** unsigned char **tagIdCount**,
unsigned char **tagId**[tagIdCount]**,** unsigned long **accessPassword,**

Return Values: RFE_RET_VALUE **status**

Status Values: RFE_RET_SUCCESS, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_COULD_NOT_WRITE, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Set the tag 30-08-33-b2-dd-d9-01-40-35-05-00-00 to ReadProtect:

PC -> Reader

52 46 45 01 5010 02 12 03 01 0C 300833b2ddd9014035050000 12345678 04 cs

dataLength	= 0x12
command	= 0x01 -> NXP-ReadProtect
tagIdCount	= 0x0C -> 12 Bytes
tagId	=30-08-33-b2-33-33-01-40-35-05-00-00
accessPassword	= 0x12345678

Reader -> PC 52 46 45 01 5010 02 01 03 00 04 cs

dataLength	= 0x01
status	= 0x00 -> RFE_RET_SUCCESS

4.2 NXP-Clear-ReadProtect (02)

This function retrieves a NXP tag from the ReadProtect Mode.

Parameters: unsigned char **command,** unsigned char **tagIdCount**,
unsigned char **tagId**[tagIdCount]**,** unsigned long **accessPassword,**

Return Values: RFE_RET_VALUE **status**

Status Values: RFE_RET_SUCCESS, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_COULD_NOT_WRITE, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Clear the ReadProtect of the tag 00-00-00-00-00-00-00-00-00-00-00-00:

PC -> Reader

52 46 45 01 5010 02 12 03 02 0C 000000000000000000000000 12345678 04 cs

dataLength	= 0x12
command	= 0x02 -> NXP-Clear-ReadProtect
tagIdCount	= 0x0C -> 12 Bytes
tagId	=00-00-00-00-00-00-00-00-00-00-00-00
accessPassword	= 0x12345678

Reader -> PC 52 46 45 01 5010 02 01 03 00 04 cs

dataLength	= 0x01
status	= 0x00 -> RFE_RET_SUCCESS

4.3 Block-Write-To-Tag (03)

This function tries to execute a Gen2 Block-Write command to the tag.

The data which is sent to the reader is split up into (**byteCount**/2)/**blockSize_word** blocks, that are each sent with the Gen2 command Block-Write. If the last block consists of less words than **blockSize_word**, the block is sent with this lower block size.

If the tag can handle a Block-Write command and with which block size, can be found in the tags manual.

Parameters: unsigned char **tagIdCount**, unsigned char **tagId**[tagIdCount]**,**
unsigned char **memoryBank,** unsigned short **startAddress**,
unsigned long **accessPassword,** unsigned char **blockSize_word**,
unsigned char **byteCount,** unsigned long **accessPassword,**

Return Values: RFE_RET_VALUE **status**

Status Values: RFE_RET_SUCCESS, RFE_RET_RESULT_PENDING, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Write 5 byte to the tag 30-08-33-b2-dd-d9-01-40-35-05-00-00 at the memory bank 1 and the start address 0x12:

PC -> Reader

52 46 45 01 5010 02 1E 03 03 0C 300833b2ddd901403550000 01 0000 00000000 01
06 020023A488C 04 cs

dataLength	= 0x1E
tagIdCount	= 0x0C -> 12 Bytes
tagId	=30-08-33-b2-33-33-01-40-35-05-00-00
memoryBank	= 0x01 -> second bank
startAdress	= 0x0000
accessPassword	= 0x00000000
blockSize_word	= 0x02 -> 2 words are sent per block
bytesCound	= 0x06 -> 6 Bytes
data	= 0x020023A488C

Reader -> PC 52 46 45 01 5010 02 01 03 00 04 cs

dataLength	= 0x01
status	= 0x00 -> RFE_RET_SUCCESS

In this example, the data to write is split up into 2 blocks. The first block consists of the specified blockSize: 4 Bytes /2 Words. The last block consists only of the remaining 2 Bytes / 1 Word.

4.4 Get-Handle-From-Tag (04)

This function tries to get the handle of a specified tag.

Parameters: unsigned char **command,** unsigned char **tagIdCount**,
unsigned char **tagId**[tagIdCount]

Return Values: RFE_RET_VALUE **status**, unsigned char **handle**[2]

Status Values: RFE_RET_SUCCESS, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_COULD_NOT_WRITE, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Get the handle from the tag 30-08-33-b2-dd-d9-01-40-35-05-00-00:

PC -> Reader

52 46 45 01 5010 02 0E 03 04 0C 300833b2ddd90140350500000 04 cs

dataLength	= 0x12
command	= 0x04 -> Get-Handle-From-Tag
tagIdCount	= 0x0C -> 12 Bytes
tagId	=30-08-33-b2-dd-d9-01-40-35-05-00-00

Reader -> PC 52 46 45 01 5010 02 01 03 00 1234 04 cs

dataLength	= 0x01
status	= 0x00 -> RFE_RET_SUCCESS
handle	= 12-34

4.5 Read-From-Handle (05)

With this function data can be read from the memory of a tag via the specified handle.

Parameters: unsigned char **handle**[2]**,** unsigned char **memoryBank,** unsigned short **startAddress**, unsigned long **accessPassword,** unsigned char **byteCount**

Return Values: RFE_RET_VALUE **status,** unsigned char **byteCount,** unsigned char **data[byteCount]**

Status Values: RFE_RET_SUCCESS, RFE_RET_RESULT_PENDING, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Read 5 byte from the tag with the handle 12-34 at the memory bank 1 and the start address 0x12:

PC -> Reader

52 46 45 01 5010 02 0B 03 05 1234 01 0000 00000000 06 04 cs

dataLength	= 0x0B
command	= 0x05 -> Read-From-Handle
handle	= 12-34
memoryBank	= 0x01 -> second bank
startAddress	= 0x0000
accessPassword	= 0x00000000
bytesCount	= 0x06 -> 6 Bytes

Reader -> PC 52 46 45 01 5010 02 08 03 00 06 020023A4884C 04 cs

dataLength	= 0x08
status	= 0x00 -> RFE_RET_SUCCESS
bytesCount	= 0x06 -> 6 Bytes
data	= 0x020023A4884C

4.6 Write-To-Handle (06)

With this function data can be written to the memory of a tag via the specified handle.

Parameters: unsigned char **handle**[2]**,** unsigned char **memoryBank,** unsigned short **startAddress**, unsigned long **accessPassword,** unsigned char **byteCount**[byteCount]

Return Values: RFE_RET_VALUE **status**

Status Values: RFE_RET_SUCCESS, RFE_RET_RESULT_PENDING, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_COULD_NOT_WRITE, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Write 5 byte to the tag with the handle 12-34 at the memory bank 1 and the start address 0x12:

PC -> Reader

52 46 45 01 5010 02 11 03 06 1234 01 0000 00000000 06 020023A4884C 04 cs

dataLength	= 0x11
command	= 0x06 -> Write-To-Handle
handle	= 12-34
memoryBank	= 0x01 -> second bank
startAddress	= 0x0000
accessPassword	= 0x00000000
bytesCount	= 0x06 -> 6 Bytes
data	= 0x020023A4884C

Reader -> PC 52 46 45 01 5010 02 01 03 00 04 cs

dataLength	= 0x08
status	= 0x00 -> RFE_RET_SUCCESS

4.7 Custom-tag-Command-Handle (07)

This function enables the reader to send custom Gen2 commands to the tag.

If the password is not zero, the reader tries to access the tag. The reader always appends the 16-bit handle at the end of the command.

Parameters: unsigned char **handle**[2]**,** unsigned char **command**[2],
unsigned long **accessPassword,** unsigned char **txBitCount**,
unsigned long **txBits** [bitCount*8]

Return Values: RFE_RET_VALUE **status**, bool **headerBit**,
unsigned char **rxByteCount**, unsigned char **rxBytes** [rxByteCount]

Status Values: RFE_RET_SUCCESS, RFE_RET_RESULT_PENDING, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_COULD_NOT_WRITE, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Send the custom command E0-01 with 9 bits payload (100111001) to the tag:

PC -> Reader

52 46 45 01 5010 02 0C 03 07 1234 E001 00000000 09 9C80 04 cs

dataLength	= 0x0C
command	= 0x07 -> Custom-Tag-Command-Handle
handle	= 12-34
command	= 0xE001
accessPassword	= 0x00000000
txBitCount	= 0x09 -> 9 bits
txBits	= 0x9C90 -> 100111001

Reader -> PC 52 46 45 01 5010 02 07 03 00 00 06 02001234 04 cs

dataLength	= 0x08
status	= 0x00 -> RFE_RET_SUCCESS
headerBit	= 0x00 -> No header bit
bytesCount	= 0x04 -> 4 Bytes
data	= 0x02001234

4.8 Block-Write-To-Tag (08)

This function tries to execute a Gen2 Block-Write command to the tag via th specified handle.

The data which is sent to the reader is split up into (**byteCount**/2)/**blockSize_word** blocks, that are each sent with the Gen2 command Block-Write. If the last block consists of less words than **blockSize_word**, the block is sent with this lower block size.

If the tag can handle a Block-Write command and with which block size, can be found in the tags manual.

Parameters: unsigned char **handle**[2], unsigned char **memoryBank,**
unsigned short **startAddress**, unsigned long **accessPassword,**
unsigned char **blockSize_word**, unsigned char **byteCount,**
unsigned char **data**[byteCound]

Return Values: RFE_RET_VALUE **status**

Status Values: RFE_RET_SUCCESS, RFE_RET_RESULT_PENDING, RFE_RET_ERR_ON_EXEC_OP, RFE_RET_ERR_WRONG_PARAM_COUNT, RFE_RET_ERR_WRONG_PARAM, Every TMI Return Code

Example: Write 5 byte to the tag 30-08-33-b2-dd-d9-01-40-35-05-00-00 at the memory bank 1 and the start address 0x12:

PC -> Reader

52 46 45 01 5010 02 12 03 03 1234 01 0000 00000000 01 06 020023A488C 04 cs

dataLength	= 0x12
handle	= 12-34
memoryBank	= 0x01 -> second bank
startAdress	= 0x0000
accessPassword	= 0x00000000
blockSize_word	= 0x02 -> 2 words are sent per block
bytesCound	= 0x06 -> 6 Bytes
data	= 0x020023A488C

Reader -> PC 52 46 45 01 5010 02 01 03 00 04 cs

dataLength	= 0x01
status	= 0x00 -> RFE_RET_SUCCESS

In this example, the data to write is split up into 2 blocks. The first block consists of the specified blockSize: 4 Bytes /2 Words. The last block consists only of the remaining 2 Bytes / 1 Word.

5 Cyclic-Inventory Start Bytes

The standard start bytes for cyclic inventory information are extended with this type of reader. This is the complete list of start bytes.

Name	Byte	Size	Description
RFE_TAG_ID_START_BYTE	0x01	Variable	The ID of the detected Tag. The variable size of the id is sent in the first byte of the ID.
RFE_RSSI_START_BYTE	0x02	2 Bytes	The RSSI value of the detected tag.
RFE_MEMORY_START_BYTE	0x03	4 Bytes + variable	Read memory data of the tag: 1 Byte: Memory Bank 2 Bytes: Address 1 Byte: Size N Bytes: Data
RFE_ANTENNA_ID_START_BYTE	0x05	1 Byte	The antenna at which the tag was detected.
RFE_READ_FREQU_START_BYTE	0x06	3 Bytes	The frequency at which the tag was detected.
RFE_GEN2_HANDLE_START_BYTE	0x07	2 Bytes	The handle of the detected Gen2 tag.
RFE_GEN2_PC_START_BYTE	0x0A	2 Bytes	The PC of the detected Gen2 tag.
RFE_APPLICATION_START_BYTE	0x70	Variable	Additional information of the running application

Example:

52 46 45 01 9002 02 1b 03 01 0c 000000000000000000000004 02 081e 03 0200000230C2 06 0d37fc 07 5eac 0a 3000 70 02 457A 04 cs

Part	Field	Description
01 0c 000000000000000000000004	ID	ID(12 Byte): 00-00-00-00-00-00-00-00-00-00-00-04
02 081e	RSSI	Signal Q: 8 dB Signal I: 30 dB
03 02 0000 02 30C2	MEM	02: Memory Bank 0x02 0000: Address 0x0000 02: 2 Bytes of data 30C2: Read data
06 0d37fc	FREQU	Frequency: 866300 MHz
07 5eac	HANDLE	Gen2 Handle: 0x5eac
0a 3000	PC	Gen2 PC: 0x3000
70 02 457A	APPL	2 Bytes Information from application: 0x457A

6 Notifications

The available notifications are:

ID	Name	Value	Description
0	Antenna-Power-Changed	1 byte = (bool) on	The notification is sent every time the antenna power changes.
1	Frequency-Changed	3 byte = (long) frequency	The notification is sent every time the frequency changes.
2	Inventory-Round-Ended	0 byte	The notification is sent every time an inventory round ended.
3	LBT-RSSI-Value-Measured	2 byte = (short) value	The notification is sent every time the LBT implementations measured a new RSSI value.