READER-HOST-PROTOCOL

PUR - EXTENSION
**project title:** Reader-Host-Protocol - PUR-Extension

**History of Change:**

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Change Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.03.2010</td>
<td>sde</td>
<td>Created document</td>
<td>v0.01</td>
</tr>
<tr>
<td>21.06.2011</td>
<td>sde</td>
<td>Added Lock-Tag codes</td>
<td>v0.02</td>
</tr>
<tr>
<td>08.12.2011</td>
<td>sde</td>
<td>Added Gen2-EPC-Size, Added description for Read-From-Tag, Added notification values</td>
<td>v0.03</td>
</tr>
<tr>
<td>12.12.2012</td>
<td>sde</td>
<td>Added Cyclic-Inventory Start Bytes, Added Gen2-Send-Handle, Added Gen2-Send-PC</td>
<td>v0.04</td>
</tr>
<tr>
<td>09.01.2013</td>
<td>sde</td>
<td>Added Commands:</td>
<td>v0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Get-Handle-From-Tag</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Read-From-Handle</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Write-To-Handle</td>
<td></td>
</tr>
<tr>
<td>15.01.2013</td>
<td>sde</td>
<td>Added new Inventory Mode, Added new Tag-Handler</td>
<td>v0.06</td>
</tr>
<tr>
<td>02.05.2013</td>
<td>sde</td>
<td>Added Gen2 Settings:</td>
<td>v0.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Gen2-Q-Setting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Gen2-Q-Method</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Gen2-Session</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Gen2-InventoryCount</td>
<td></td>
</tr>
<tr>
<td>10.10.2013</td>
<td>sde</td>
<td>Added Custom-Tag-Command</td>
<td>v0.08</td>
</tr>
<tr>
<td>26.06.2014</td>
<td>sde</td>
<td>Added Gen2 Setting: Selection Mask</td>
<td>v0.09</td>
</tr>
<tr>
<td>28.07.2014</td>
<td>sde</td>
<td>Added Gen2-BlockWrite and PostDetect-READ</td>
<td>v0.10</td>
</tr>
<tr>
<td>22.01.2015</td>
<td>sde</td>
<td>Added some optional parameters</td>
<td>v0.11</td>
</tr>
<tr>
<td>19.03.2015</td>
<td>sde</td>
<td>Added read block size parameter</td>
<td>v0.12</td>
</tr>
<tr>
<td>21.08.2017</td>
<td>sde</td>
<td>Added Direct-Command</td>
<td>v0.13</td>
</tr>
<tr>
<td>15.03.2018</td>
<td>sde</td>
<td>Added Error Codes</td>
<td>v0.14</td>
</tr>
</tbody>
</table>
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 Send Frequency ................................................................................................. 9
   2.6 PostDetect-READ ............................................................................................ 10
   2.7 Gen2-Link-Frequency ....................................................................................... 11
   2.8 Gen2-Bit-Encoding ............................................................................................ 11
   2.9 Gen2-Modulation Depth ................................................................................... 12
   2.10 Gen2-EPC-Size ............................................................................................... 12
   2.11 Gen2-Send-Handle .......................................................................................... 13
   2.12 Gen2-Send-PC ................................................................................................. 13
   2.13 Gen2-Q-Setting ............................................................................................... 13
   2.14 Gen2-Q-Method ............................................................................................... 14
   2.15 Gen2-Session ................................................................................................... 14
   2.16 Gen2-InventoryCount ...................................................................................... 14
   2.17 Gen2-Select-Mask #1 ...................................................................................... 15
   2.18 Gen2-Select-Mask #1 ...................................................................................... 15
   2.19 Gen2-Read-Block-Size ................................................................................... 16
   2.20 Gen2-Use-TText .............................................................................................. 16

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 Block-Write-To-Tag (03) .................................................................................................................. 21
4.4 Get-Handle-From-Tag (04) ............................................................................................................... 22
4.5 Read-From-Handle (05) .................................................................................................................... 23
4.6 Write-To-Handle (06) ......................................................................................................................... 24
4.7 Custom-Tag-Command-Handle (07) ................................................................................................. 25
4.8 Block-Write-To-Handle (08) .............................................................................................................. 26
4.9 Direct-Command (09) ......................................................................................................................... 27

5 Cyclic–Inventory Start Bytes........................................................................................................... 28

6 Notifications ..................................................................................................................................... 30

7 Error Codes....................................................................................................................................... 30
1 Introduction

This document describes the extensions to the standard RF-Embedded 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
<th>Size</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory Mode</td>
<td>0x0000</td>
<td>1 Byte</td>
<td>The used inventory mode</td>
<td></td>
</tr>
<tr>
<td>Power Safe Settings</td>
<td>0x0001</td>
<td>3 Bytes</td>
<td>The settings for the power safe</td>
<td></td>
</tr>
<tr>
<td>RSSI</td>
<td>0x0002</td>
<td>1 Byte</td>
<td>Defines if the RSSI Value is sent to the host</td>
<td></td>
</tr>
<tr>
<td>Tag-Id-Behavior-Mode</td>
<td>0x0003</td>
<td>1 Byte</td>
<td>Defines how the reader should behave if a tag is detected.</td>
<td></td>
</tr>
<tr>
<td>Send-Frequency</td>
<td>0x0005</td>
<td>1 Byte</td>
<td>Defines if the reader should send the frequency, where a tag was detected.</td>
<td>v1.09</td>
</tr>
<tr>
<td>PostDetect-READ</td>
<td>0x0006</td>
<td>9 Bytes</td>
<td>Settings for a read after a tag was detected.</td>
<td>v2.07</td>
</tr>
<tr>
<td>Gen2-Link-Frequency</td>
<td>0x0020</td>
<td>1 Byte</td>
<td>The used link frequency</td>
<td></td>
</tr>
<tr>
<td>Gen2-Bit-Encoding</td>
<td>0x0021</td>
<td>1 Byte</td>
<td>The used bit encoding</td>
<td></td>
</tr>
<tr>
<td>Gen2-Modulation-Depth</td>
<td>0x0022</td>
<td>1 Byte</td>
<td>The used modulation depth</td>
<td></td>
</tr>
<tr>
<td>Gen2-EPC-Size</td>
<td>0x0023</td>
<td>1 Byte</td>
<td>The expected EPC Size</td>
<td>v1.07</td>
</tr>
<tr>
<td>Gen2-Send-Handle</td>
<td>0x0024</td>
<td>1 Byte</td>
<td>Send Handle with EPC</td>
<td>v1.17</td>
</tr>
<tr>
<td>Gen2-Send-PC</td>
<td>0x0025</td>
<td>1 Byte</td>
<td>Send PC with EPC</td>
<td>v1.17</td>
</tr>
<tr>
<td>Gen2-Q-Setting</td>
<td>0x0026</td>
<td>3 Byte</td>
<td>Q settings for the inventory</td>
<td>v2.01</td>
</tr>
<tr>
<td>Gen2-Q-Method</td>
<td>0x0027</td>
<td>1 Byte</td>
<td>Q adjust method</td>
<td>v2.01</td>
</tr>
<tr>
<td>Gen2-Session</td>
<td>0x0028</td>
<td>1 Byte</td>
<td>Gen2 session</td>
<td>v2.01</td>
</tr>
<tr>
<td>Gen2-InventroyCount</td>
<td>0x0029</td>
<td>1 Byte</td>
<td>Inventory Count</td>
<td>v2.01</td>
</tr>
<tr>
<td>Gen2-Select-Mask #1</td>
<td>0x002A</td>
<td>6 Byte</td>
<td>Selection Mask #1 used for Gen2-Select command</td>
<td>v2.06</td>
</tr>
<tr>
<td>Gen2-Select-Mask #2</td>
<td>0x002B</td>
<td>6 Byte</td>
<td>Selection Mask #2 used for Gen2-Select command</td>
<td>v2.06</td>
</tr>
<tr>
<td>Gen2-Read-Block-Size</td>
<td>0x002C</td>
<td>1 Byte</td>
<td>Count of words at Gen2 Read operation</td>
<td>v2.11</td>
</tr>
<tr>
<td>Gen2-Use-TRext</td>
<td>0x002D</td>
<td>1 Byte</td>
<td>Defines if TRext should be used.</td>
<td>v2.19</td>
</tr>
</tbody>
</table>
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 \[\text{Inventory-Mode-Enum}\]

**Possible settings:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen2 - Fast Multi Tag</td>
<td>0x00</td>
<td>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.</td>
</tr>
<tr>
<td>Gen2 - Fast Single Tag</td>
<td>0x01</td>
<td>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</td>
</tr>
<tr>
<td>Gen2 - Standard Multi Tag</td>
<td>0x02</td>
<td>Inventory mode like defined in the standard</td>
</tr>
<tr>
<td>Gen2 - NXP-ReadProtect Inventory</td>
<td>0x03</td>
<td>Inventory that only searches for read protected tags. The epc of these tags will always be composed of zeros.</td>
</tr>
<tr>
<td>Gen2 - Standard Single Tag</td>
<td>0x04</td>
<td>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</td>
</tr>
</tbody>
</table>

**Default:**

Inventory-Mode-Enum 0x02 Gen2 – Standard Multi Tag
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:**

<table>
<thead>
<tr>
<th>switch</th>
<th>sleep time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x00FA</td>
</tr>
<tr>
<td>OFF</td>
<td>250 ms</td>
</tr>
</tbody>
</table>

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:**

<table>
<thead>
<tr>
<th>switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
</tr>
<tr>
<td>OFF</td>
</tr>
</tbody>
</table>
2.4 Tag-Id-Behavior

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

Data structure:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send-Tag-Id-Immediately</td>
<td>0x00</td>
<td>Every time the reader detects a tag, it is immediately forwarded to the host.</td>
</tr>
<tr>
<td>Send-Tag-Id-Once</td>
<td>0x01</td>
<td>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.</td>
</tr>
<tr>
<td>Send-Tag-Id-Immediately And Stop</td>
<td>0x02</td>
<td>The same Tag-id behavior like Send-Tag-Id-Immediately but when the first tag is detected, the Cyclic-Inventory is stopped.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>0x01 / 0x00</td>
<td>OFF (0x00)</td>
</tr>
</tbody>
</table>

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 byte Address
- 1 byte Byte-Count to Read
- 4 byte 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:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>40 kHz</td>
</tr>
<tr>
<td>0x01</td>
<td>80 kHz</td>
</tr>
<tr>
<td>0x02</td>
<td>160 kHz</td>
</tr>
<tr>
<td>0x03</td>
<td>213 kHz</td>
</tr>
<tr>
<td>0x04</td>
<td>256 kHz</td>
</tr>
<tr>
<td>0x05</td>
<td>320 kHz</td>
</tr>
</tbody>
</table>

**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:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>FM0</td>
</tr>
<tr>
<td>0x01</td>
<td>Miller 2</td>
</tr>
<tr>
<td>0x02</td>
<td>Miller 4</td>
</tr>
<tr>
<td>0x03</td>
<td>Miller 8</td>
</tr>
</tbody>
</table>

**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:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Dynamic</td>
</tr>
<tr>
<td>2</td>
<td>2 Byte EPC</td>
</tr>
<tr>
<td>4</td>
<td>4 Byte EPC</td>
</tr>
<tr>
<td>12</td>
<td>12 Byte EPC</td>
</tr>
<tr>
<td>18</td>
<td>18 Byte EPC</td>
</tr>
</tbody>
</table>

**Default:**

| EPC Size | 12 | Constant 12 Byte 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 $2^Q$.

**Data Structure:**

- 1 byte **Initial Q**
- 1 byte **Minimal Q**
- 1 byte **Maximal Q**

**Default:**

<table>
<thead>
<tr>
<th>Q</th>
<th>Initial Q</th>
<th>Minimal Q</th>
<th>Maximal Q</th>
<th>Slot Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>32768</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Static (Q stays at initial value)</td>
</tr>
<tr>
<td>0x01</td>
<td>RFE Dynamic Adjust Method</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Session 0</td>
</tr>
<tr>
<td>0x01</td>
<td>Session 1</td>
</tr>
<tr>
<td>0x02</td>
<td>Session 2</td>
</tr>
<tr>
<td>0x03</td>
<td>Session 3</td>
</tr>
</tbody>
</table>

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:
Count 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 byte 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

**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 Bit
- 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 byte 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.

2.20 Gen2-Use-TRext
Defines if the TRext should be used. The selected mode is transmitted to the tag while issuing a QUERY command. No TRext is only supported with Miller4 and Miller8

Data Structure:

1 byte Use-TRext

Default:

Use-TRext 0x01 TRext is used always
3 Function Descriptions

3.1 Read-From-Tag (50-03)
If the byte size 0 is selected, the reader reads until the tag responds 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:

```c
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 their 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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NXP-Set-ReadProtect</td>
<td>0x01</td>
</tr>
<tr>
<td>NXP-Clear-ReadProtect</td>
<td>0x02</td>
</tr>
<tr>
<td>Block-Write-To-Tag</td>
<td>0x03</td>
</tr>
<tr>
<td>Get-Handle-From-Tag</td>
<td>0x04</td>
</tr>
<tr>
<td>Read-From-Handle</td>
<td>0x05</td>
</tr>
<tr>
<td>Write-To-Handle</td>
<td>0x06</td>
</tr>
<tr>
<td>Custom-Tag-Command-Handle</td>
<td>0x07</td>
</tr>
<tr>
<td>Block-Write-To-Handle</td>
<td>0x08</td>
</tr>
</tbody>
</table>
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,
un signed 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 00-00-00-00-00-00-00-00-00-00-00-00-00-00:

PC -> Reader

Reader -> PC
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 \((\text{byteCount}/2)/\text{blockSize word}\) 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 \(\text{tagIdCount}\)
- unsigned char \(\text{tagId}[\text{tagIdCount}]\)
- unsigned char \(\text{memoryBank}\)
- unsigned short \(\text{startAddress}\)
- unsigned long \(\text{accessPassword}\)
- unsigned char \(\text{blockSize word}\)
- unsigned char \(\text{byteCount}\)
- unsigned char \(\text{data}[\text{byteCount}]\)

**Return Values:** RFE_RET_VALUE \(\text{status}\)

**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 50 10 02 0E 03 0C 30 08 33 b2 d6 d9 01 40 35 05 00 00 01 00 00 00 00 00 00 00 00 01 06 02 00 23 A4 88 4C
```

- **dataLength** = 0x1E
- **tagIdCount** = 0x0C - 12 Bytes
- **tagId** = 30-08-33-b2-33-33-01-40-35-05-00-00
- **memoryBank** = 0x01 - second bank
- **startAddress** = 0x0000
- **accessPassword** = 0x00000000
- **blockSize_word** = 0x02 - 2 words are sent per block
- **byteCount** = 0x06 - 6 Bytes
- **data** = 0x020023A4884C

Reader -> PC

```
52 46 45 01 50 10 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 consist 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 300833b2ddd9014035050000 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 03 03 00 1234 04 cs
```

dataLength = 0x03
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,

Optional: [ unsigned char retryOnError (!= 0) ]

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_COULD_NOT_WRITE,
- 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

\[
\begin{array}{cccccccc}
52 & 46 & 45 & 01 & 50 & 10 & 02 & 08 & 03 & 05 & 1234 & 01 & 0000 & 00000000 & 06 & 04 & \text{cs}
\end{array}
\]

\[
\begin{array}{l}
\text{dataLength} = 0x0B \\
\text{command} = 0x05 -> \text{Read-From-Handle} \\
\text{handle} = 12-34 \\
\text{memoryBank} = 0x01 -> \text{second bank} \\
\text{startAddress} = 0x0000 \\
\text{accessPassword} = 0x00000000 \\
\text{bytesCount} = 0x06 -> 6 \text{Bytes}
\end{array}
\]

Reader -> PC

\[
\begin{array}{cccccccc}
52 & 46 & 45 & 01 & 50 & 10 & 02 & 08 & 03 & 06 & 020023A4884C & 04 & \text{cs}
\end{array}
\]

\[
\begin{array}{l}
\text{dataLength} = 0x08 \\
\text{status} = 0x00 -> \text{RFE_RET_SUCCESS} \\
\text{bytesCount} = 0x06 -> 6 \text{Bytes} \\
\text{data} = 0x020023A4884C
\end{array}
\]
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`, `unsigned char data [byteCount]`

**Optional:**
- `[ unsigned char retryOnError (!= 0), unsigned char verifyReadOnError (!= 0) ]`

**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 00 020023A4884C 04 cs
```

- `dataLength` = 0x11
- `command` = 0x06 -> Write-To-Handle
- `handle` = 12-34
- `memoryBank` = 0x01 -> first 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` = 0x01
- `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 long accessPassword`, `unsigned char txBitCount`,
- `unsigned char txBits [bitCount*8]`, `unsigned char estimatedRxBitCount`

**Optional:**
- `[unsigned short estimatedTagProcessingTime, unsigned char retryOnError (!= 0)]`

**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 00000000 09 9C80 31 04 cs
```

- `dataLength` = 0x0C
- `command` = 0x07 -> Custom-Tag-Command-Handle
- `handle` = 12-34
- `command` = 0xE001
- `accessPassword` = 0x00000000
- `txBitCount` = 0x09 -> 9 Bits
- `txBits` = 0x9C80 -> 100111001
- `estimatedRxBitCount` = 0x31 -> 49 Bits

```
Reader -> PC
52 46 45 01 5010 02 07 03 00 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-Handle (08)

This function tries to execute a Gen2 Block-Write command to the tag via the specified handle. The data which is sent to the reader is split up into \( \text{byteCount}/2 \)/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 [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_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:

```plaintext
Reader -> PC:
52 46 45 01 5010 02 12 03 03 1234 01 0000 00000000 01
  00 020023A4884C 04 cs
```

- dataLength = 0x12
- handle = 12-34
- memoryBank = 0x01 -> second bank
- startAddress = 0x0000
- accessPassword = 0x00000000
- blockSize_word = 0x02 -> 2 words are sent per block
- byteCount = 0x06 -> 6 Bytes
- data = 0x020023A4884C

```plaintext
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 consist of the specified block size: 4 Bytes /2 Words. The last block consists only of the remaining 2 Bytes / 1 Word.
4.9 Direct-Command (09)

This function enables the reader to send a direct command to the tag. If estimated RX Bit count is zero, the reader does not wait for any tag reply.

Parameters: bool sendWithCRC, bool sendWithTRCal, unsigned short txBitCount, unsigned char txBits [bitCount*8], unsigned short estimatedRxBitCount

Optional: [unsigned short estimatedTagProcessingTime, unsigned char retryOnError (!= 0)]

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 direct command C2-80-00-E6-C5-40 with 42 bits to the tag:

PC -> Reader

52 46 45 01 5010 02 0D 03 09 01 00 02 0A C28000E6C540 0031 04 cs

dataLength = 0x0D
command = 0x09 -> Direct-Command
sendWithCRC = 0x01 -> true
sendWithTRCal = 0x00 -> false

xbits = 0x002A -> 42 Bits

estimatedRxBitCount = 0x0031 -> 49 Bits

Reader -> PC

52 46 45 01 5010 02 0B 03 00 00 08 E280B0443C009B15 04 cs

dataLength = 0x08
status = 0x00 -> RFE_RET_SUCCESS
headerBit = 0x00 -> No header bit
bytesCount = 0x08 -> 8 Bytes
data = 0xE280B0443C009B15
## 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.

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

```
52 46 45 01 9002 02 26 03 01 0c 000000000000000000000004 02 081e 03 020000230C2 06 0d37fc 07 5ead 0a 3000 70 02 457A 04 cs
```

<table>
<thead>
<tr>
<th>Part</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0c</td>
<td>ID((0x0C = 12 \text{ Byte}): 00-00-00-00-00-00-00-00-00-00-00-00-04)</td>
</tr>
<tr>
<td>02</td>
<td>081e</td>
<td>RSSI (\text{ Signal Q: 8dB, Signal I: 30dB})</td>
</tr>
<tr>
<td>03</td>
<td>02 0000 02 30C2</td>
<td>MEM (\text{02: Memory Bank 0x02 0000: Address 0x0000 02: 2 Bytes of data 30C2: Read data})</td>
</tr>
<tr>
<td>06</td>
<td>0d37fc</td>
<td>FREQU (\text{Frequency: 866300 MHz})</td>
</tr>
<tr>
<td>07</td>
<td>5ead</td>
<td>HANDLE (\text{Gen2 Handle: 0x5ead})</td>
</tr>
<tr>
<td>0a</td>
<td>3000</td>
<td>PC (\text{Gen2 PC: 0x3000})</td>
</tr>
<tr>
<td>70</td>
<td>02 457A</td>
<td>APPL (\text{2 Bytes Information from application: 0x457A})</td>
</tr>
</tbody>
</table>
6 Notifications

The available notifications are:

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

7 Error Codes

Theses error codes are used with the Error-Occurred-Interrupt (09-FF):

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>RFP-Too-High</td>
<td>1 byte = (uchar) antennaId</td>
<td>If automatic reflected power measurement is activated, the reader measures the RFP every time the antenna should be activated. If the measured value is above the set threshold, this error code is sent. While the antenna should be active, the measurement is retried constantly, until the either the value is below the threshold, or the antenna should be switched off.</td>
</tr>
</tbody>
</table>