# User Manual
# Indy 2k UHF Reader
# For
# Muehlbauer RFID production machines

Version 1.8

Indy 2k UHF Reader

Indy 2k UHF reader electronic

## Index

## Safety instructions

⇨ **FCC:**

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) this device may not cause harmful interferences, and
(2) this device must accept any interference received, including interference that may cause undesired operation.

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

The digital part of this device is authorized under the Supplier's Declaration of Conformity (SDoC) procedure.

The Federal Communications Commission (FCC) warns the users that changes or modifications to the unit not expressly approved by the party responsible for compliance could void the user’s authority to operate the equipment.

⇨ **RF exposure statement (mobile and fixed devices):**

This device complies with the RF exposure requirements for mobile and fixed devices. However, the device shall be used in such a manner that the potential for human contact during normal operation is minimized.

The system is power up with 24 volts external DC power.
Please carefully when you connect the power wires. Connect wires only when power supply is switched off!
In other case you can create shorted circuits and lightning’s when you connect under present power!

Internal reader systems run with 5V, 3.3V and 1,8V DC power.

# 1 System Overview

The process department of Muehlbauer develop a new UHF reader hardware together with Impinj Inc. USA. Basic component of this system are the Impinj IPJ-E3004 reader board from Indy@R2000 development platform. This reader board provide all features of INDY 2000 reader chip family and support all UHF reader features of standard EPC1 Gen2 V 1.2. This reader chipset represent 70% of high performance readers in the RFID market and Impinj are here technology leader.

The Indy 2k reader provide a lot of functions und analyse tools for RFID processes. We can describe here only the basic functions for our machine applications.
Impinj provide for that reader board an API SDK. All tools and functions of these SDK have a high intelligence and create automatic sequences, but exact at this point was the problem for us. The DLL's use often complex functions and commands in automatic routines and for the user is it very complicated to know the actual setup of the system. The API functions change often the hardware configuration and the User are not shure which register have now which value.
We test this SDK system over some weeks and finally we come to the conclusion, it create a lot of confusion and you need more time, to analyse why the system do not what you expected.

With the good manual for the low level communication between reader controller and HOST was possible for us, to understand the function of config registers, to extract the important commands and select the sequences for complex programming cycles for RFID chips.
With this knowledge are possible for us to implement low level reader commands directly into applications.
We start the same process with Thinkmagic's M6e reader module, but here are no low level command language or documentation available and we stop this.

## 1.1 How is the reader hardware used

The Muehlbauer Company built production machines for RFID Labels, hangtags and cards with RFID functionalities. In our production machines are test equipment required for check the production quality and find no functional products at end of production line. The UHF reader in this document are used to test the products of the machines and Read or Program data into the integrated RFID chips.

The UHF reader hardware are part of the production machine for RFID self-adhesive labels and paper tickets with RFID functionality. This reader will not sale as standalone unit for other applications and the hardware are every time part of a production machine or machine family in industrial environment.

The reader case are installed inside machine cabinet and the antenna are connected over 2 – 3 meters RG316 coaxial cable with reader hardware. Antenna are located in production flow to "see" the produced RFID transponders at end of the machine for a functional test. If a product not functional, we make that with ink jet printers.

The reader system gets electrical power over 24V power supplies from machine side and is connected to ground potential of the machine base frame. Control commands, setup parameters and data come from machine software and the reader hardware send the results back to machine control units like SPS or PC equipment.

The configuration of reader hardware are depend from country regulations and we limit the used frequency range in production process to the country related frequency bands. The frequency band are depend from telecommunication rules at customer side where the machine is located.

The setup of the reader hardware works in production with low power levels (normally ca. 5 - 15dBm) and we shield the radiated power with metal plates and antennas in metal boxes. The standard reader antenna is the Miniguardrail antenna from manufacturer Impinj. This antenna type is part of the FCC certification and the data sheet is include.

# 2 Reader Firmware

The firmware of this Indy 2k reader is written from Impinj and called Indy MAC Firmware. The current version is 2.6.0. All parameters and values store into MAC memory registers and it exist different register areas for custom modifications, Macros, RFID chip functions, Calibration and programming sequences.

### 2.1.1 MAC Register

All config parameters must store in MAC registers and after that, the values are true for the MAC firmware. The MAC registers area divided in address ranges for Antenna setups, Frequency parameters, Power levels, Calibration, Test modes, Inventory, Tag Access and Special functions. The register address consist every time as 2 bytes and the detail description are in PDF document „MAC Register Descriptions”.
All configurations for Read, Write functions and Setups work over these register addresses.
Test commands must write into Command register „F0 00h“. The MAC firmware will start the function imedially and respond with different response packet structures.

### 2.1.2 OEM Register

OEM registers are used to store special customized setups and called during boot process. All parameters from OEM registers will copy to MAC registers and used for the next function. The system use every time two OEM registers, one to store the MAC address and a second register for the data value self.
The OEM register range starts with address 0C 60h.

The first OEM register 0C60h get the address of the MAC register. The follow OEM register 0C61h get the data value for this MAC register and so on.
The reboot cycle do copy the OEM information into MAC registers and customer can store different boot parameters.
We do not use the OEM register functions and send the MAC register values directly to the reader firmware. The advantage of this procedure are, after reboot, the reader have every time default values and the start situation after reader replacement or machine reboot are 100% clear.

### 2.2 Software Tools

#### 2.2.1 Indy Tool (Development Tool)

For the communication with Indy reader hardware exist from Impinj side and development tool with name "Indy Tool". The current version is 2.6.0 but the new version 2.7.0 for new MAC firmware 2.7.0 is available. Differences between both tools are, the new version 2.7.0 can handle additionally new EPC1 Gen2 functions like "Untraceable" and "Authenticate". This tool works with the SDK from Impinj and run with script files.
For basic functions is this tool okay, but for more complex tests exist problems. When you will realize different functions, you must often switch between tool sides and every side "create" own reader setup. After short time, you are not shure what is the current setup in reader, or nobody know why will this unit not work……

With these bad experience, we take our own reader tool, designed for the Skyetek M9 readers, and implement the functions for the Indy 2k reader system.

#### 2.2.2 MB-Reader-Tool for INDY 2k and Skyetek UHF Reader

The MB reader Tool was used in history only for the Skyetek M9 UHF readers and it helps to configure the reader and start test functions for Read or Write cycles.
This tool can handle Macro files and we built our command sequences for Init, Frequency setup and chip test applications.
This software tool is only available for MB employers or Service guys and partners.
Normally you need a little bit training to understand the packet structures and communication at low level serial language. But after first steps with the tool are the handling easy and you know what the reader do. This knowledge are important when you analyse error codes or Service requests.

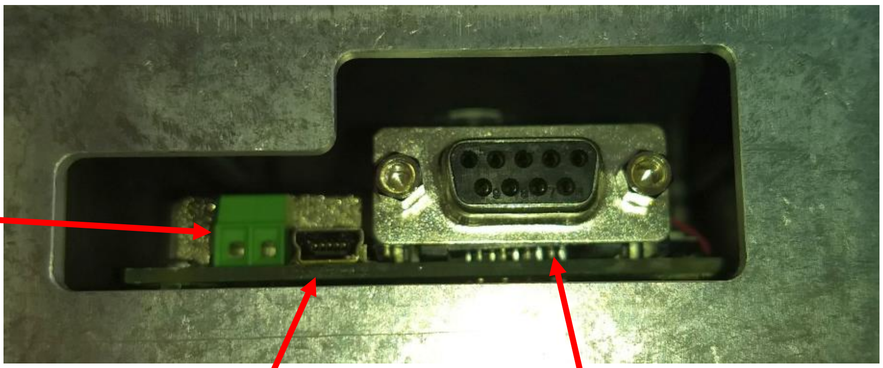
# 3 Hardware interfaces

### 3.1 Serial HOST Interface

The microcontroller at reader board is very powerful and can handle HOST data extremely fast. The UART interface run with 115,2kBit/s 8N1 and the advantage of the UART port is, you can monitor the data streams very easy.
Voltage level at UART port are +/- 12V.
The data code for this interface are hex-based and have a 8 byte structure.
All Write commands to a MAC register starts with 01h and Read commands with 00h.

⇨ ***The UART interface works as 3 wire port with TXD, RXD and Ground. Handshake wire are not required and can create problems. The best is you use only the 3-wire cable with straight pinning. Data rate is 115,2kBit/s 8N1, Hex code***

With this UART interface is the hardware easy to integrate in industrial machines and with a data spion you can monitor the command dialog for fault analysis.



### 3.2 Mini-USB interface

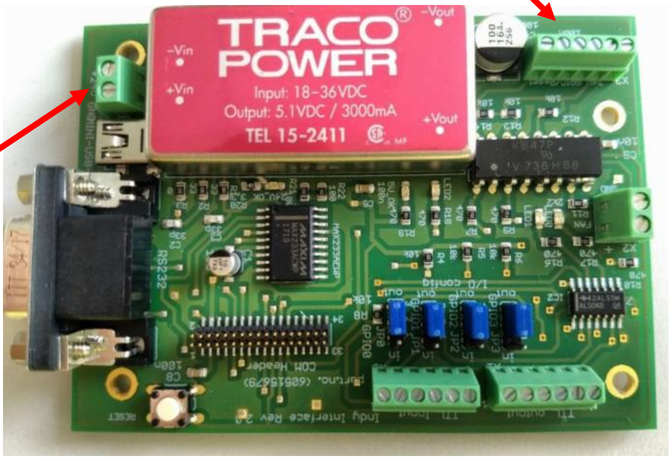
The Indy 2k UHF reader provides a USB HOST interface with Mini-USB socket. This interface is not active in default setup.
The reader can power up via USB but before you  must set two jumpers at reader board from Main power to USB power.

### 3.3 Reset input

The reader hardware have since revision 2.0 and optoisulated reset input for hardware resets. 24 volts at input start a reset when signal are min. 50ms present.

### 3.4 GPIO 24V Interface

The Indy 2k reader have three optocouplers isolated outputs. The signals are inverted coupled with the first three TTL output lines.



## 3.5 GPIO TTL Interface

The TTL GPIO interface provide four lines and can independent configure as Input or Output by jumper at interface board.
Voltage level for Inputs are max. 3.3 volts and is **not** 5V tolerant!
The TTL input lines goes directly to the ARM 7 controller of the reader board.
When you use the GPIO as output, then is a Inverter as driver between and you get a inverted logic at TTL output.
Depend from GPIO definition of the reader ports, you can control the GPIO lines by HOST commands over the ARM-7 microcontroller.

## 3.6 Power supply at Interface Bord

The reader system have an internal DC/DC power converter to convert incoming 24V DC power into 5V / 3000mA reader power.
At interface board exist two green led’s to monitor if the DC power present or not.
The connector self are designed for two wire connections and it’s from Phoenix contact.
Max wire diameter is 1,0mm2

At reader interface board exist since PCB rev 2.0 an additional connection point for 5V DC power.
This connection point is planned for a small fan. This fan is required for reader cooling when higher RF power levels necessary or the environment temperature is higher as 30C°.

## 3.7 Antenna Connections

The reader has two antenna ports with MMCX sockets from Huber & Suhner Switzerland. The first antenna port is called Ant.0 and the second Ant.1.
Antenna port impedance is 50 ohms. We use in our setup the port 0 as main port for TX and RX signals. The port 1 is normally open and unused.

The UHF reader works in combination with a Miniguardrail antenna from Impinj. We use as antenna cable a Teflon isolated cable type RG316.
Cable length are accepted up to 3,5meters.
All antenna cable installations are inside of a production machine and follow professional rules of installations.

# 4 Command dialog between Host <==> Reader

Between Indy 2k reader and HOST system exist an intensive dialog with register and packet structures. The HOST-PC send commands to store the data values into MAC register of the reader. When the HOST will start a command, then this command code must store in execute MAC register F0 00h. The reader respond then with Start- and Stop packages, show the error status, time stamps and measured values.

If data should be programmed into a chip, then works the dialog identical. The HOST send the setup and new data to special MAC registers of the reader hardware. Then HOST send test commands to register F0 00h and the reader respond with Start packet.
When you will Read or Write chip data, then the reader respond after Start packet with Inventory packet type 00 05h. This packet have the EPC data, PC Word and RSSI values include.
If a Tag Access command execute, then you get also an Tag Access packet (type 00 06h) with command Echo, Error codes and Tag data’s for read functions.

**Data structures from Host and reader base at 8 byte frame structure. Start or End sign do not exist. Single bytes or level jumps at interface lines will be interpret as data byte. If you have this problem, the reader will not understand the commands because the first wrong byte will be part of the command.**

When you get eight FF bytes as reader answer, then have the reader a lot of questions what the HOST sends.
The reason for that problem can be interrupted data transmission or power up cycles. The reader interpret that as byte and use it as leading byte for next command.

## 4.1 Serial Re-Syncronisation with Indy 2k Reader

You have two possibilities to solve this problem:

1. Hardware reset with reset button at interface board or power OFF/ON

2. Send single zero bytes in hex code and check for reader responses. The zero bytes fill up the FiFo and when 8 bytes complete, the reader will analyse that “command” and respond. If the response not the MAC firmware information, then you must send more zero bytes to shift all wrong data bytes through the input buffer. It’s possible you must send up to 16 zero bytes to get the follow response.

```
Host ==>      00 00 00 00 00 00 00 00
Reader ==>    00 00 00 00 F0 00 06 02           (MAC Version 2.6.0)
```

Please send one „zero“ byte and check for answer (20ms). You get no response, then send next zero byte. If the 8 byte structure in FiFo complete, the reader will send an answer. If a reader answer detected, you can send the normal reader commands and the communication are in “synch” mode.

***==> The effect with wrong reader response will create, when you unplug and plug-in the serial connector or after reboot of HOST hardware.***

## 4.2 Reader commands for INDY R2k Readers

The command structure are descripted in follow documents “MAC register documentation” and in “HOST packet definitions” from Impinj.
The reader respond only at executable commands. MAC register Write cycles create no response! The reason for that are, this reduce the traffic at UART interface.
The Error status can checked with Read cycle at follow MAC addresses:
*00 05h (Current Error Status)*
*00 06h (Last existing MAC Error).*

When the reader get commands, the hardware respond with different data packets. The first response are a Start packet (type 00 00h) and the End of communication have a Stop packet with type 00 01h. Between Start- and Stop packet types exist additionally packets for Inventory (type 00 05h) and Tag Access (type 00 0 6h) functions.
Start- and Stop packets built a frame around the Tag information packets.
Detail information over packet structure and data formats are in document “HOST Packet definitions” from Impinj.

## 4.3 MAC Register Config Blocks for INDY R2k Readers

MAC register addresses are split into functional address blocks for Hardware setup, Antenna configuration , Inventory, Tag functions, Diagnostic Test modes.
Detail information over MAC register blocks and functions are in document “MAC register definitions” from Impinj.
The byte order in all commands are played and starts at left side with LSB. The MSB is at right side.
MAC Write commands starts with “01h” follow by Flag and MAC address, then follow the data value.
MAC Read commands starts with “00h” and respond as identical 8 byte structure with actual register value.

- ⇨ ***For data values are the MSB at right position, the LSB are directly after MAC address. The byte order are a little bit unusual but create more speed for reader internal actions.***

**Example Byte Order for Write to MAC Register:**

0.Byte $\xrightarrow{\hspace{3cm}}$ 7.Byte

01 00 0A 01 2C F8 0D 00 (MAC register command for set frequency)

| | | |
|---|---|---|
| 0. Byte | 01 | Write into MAC register |
| 1. Byte | 00 | Flag |
| 2.-3. Byte | 01 0A | MAC Address for frequency setup |
| 4.-7. Byte | 00 0D F8 2C | Hex value for reader frequency, here 915,5MHz |

## 4.4 Register Struktur MAC Firmware

### 4.4.1 General Block (address range 0x00)

In this address range are stored the Firmware version data, Hardware data and Serial numbers of the reader hardware. These registers have Read only status and can’t overwrite.

### 4.4.2 Test Block (address range 0x01)

This range are used for hardware values of test functions. The reader provide a second setup area for test functions with antenna port, frequency, RF power and a lot of additional parameters to create a special test scenario.

### 4.4.3 Platform Control Block (address range 0x02)

In this block can be configure diagnosis and extension functions. Special Functions for Impinj chip types can configure here.
e.g. Fast-ID, Tag-Focus, Monza 4 QT extensions. A lot of diagnosis function are depend from the values in the Control register block.

### 4.4.4 Protocol Scheduler Block (address range 0x03)

In this register block can be define the LBT register setups for Europe countries. We use here the default values from Impinj.

### 4.4.5 MAC Bypass Block (address range 0x04)

In this address range can be stored register values to bypass the ARM 7 MAC firmware. All these values will write directly into the INDY@R2000 chipset, without modifications from MAC firmware. These registers are „transparent” to configure the Indy@R2000 chip for special tests.

### 4.4.6 OEM Config Block (address range 0x05)

This address range are the OEM register range for special customized reader profiles. The MAC firmware will read the OEM registers during boot process.
The function is used the store special reader profiles for frequency, RF power loops, timings, power ramps and so on.

### 4.4.7 GPIO Block (address range 0x06)

This range define the function of four GPIO lines. The GPIO ports can configure as Input or Output. But signals at GPIO inputs can’t start special test commands. The firmware do not provide that. GPIO lines can used to control multiplexers or other external hardware over the HOST interface.

### 4.4.8 Antenna Block (address range 0x07)

This MAC register range are used for config parameters of antenna ports and multiplexer modes. Here are defined, which port are active for TX or RX and when should switch from one port to the other. The time slot for an Inventory cycle and the Inventory counter to stop the Inventory round after founded Tag will defined here.
These registers are very important and have extreme strong effects for reader function.

### 4.4.9 Tag Select Block (address range 0x08)

MAC registers in address range 08h controls the Tag Select functions. Values in this registers define the “AutoSelect” function, Query sequence and Tag select filters for EPC and TID banks.
These functions are very important for Auto Select modes in Perso machines like PL light.

### 4.4.10 Inventory Block (address range 0x09)

This address range define how the reader work during Inventory cycles. The registers define parameters for Query, Query Rep, Target A/B flip and Inventory algorithms. All parameters in this area work together with Tag Select Block 0x08 and Antenna Block 0x07.

==> *One important parameter for us are the “Inventory counter”.*

We set this counter to 0x01 to define Stop after first founded Tag during Inventory round.
With this setup, the reader will never detect more as one tag in one Inventory round and we get the data for one requested tag. In combination with EPC / TID match filters in Select Block, can you select which tag will respond in the Inventory round.

### 4.4.11 Tag Access Block (address range 0x0A)

This address range define all Tag Access functions for Read and Write cycles with RFID chips. We use this range to define the memory bank, offset and size of data inside chip memory and to send the data to reader.
The reader have special registers in this range to get new data from HOST, store it internal and use it for Write cycles.

### 4.4.12 RFTC Block (address range 0x0B)

This range store calibration values and hardware parameters of reader hardware.
We use the factory calibration values and will not modify these parameters.

### 4.4.13 Frequency Block (address range 0x0C)

This range are not used for setup reader frequencies, it’s for parameters of PLL systems, synthesizers, frequency divider factors and BIAS parameter for power amplifier.
In this area are no modification required.

### 4.4.14 Reserved Customer Block (address range 0x0F)

This address range are free and reserved for custom programs.
We do not use it at moment.

### 4.4.15 Host Command Block (address range 0xF0)

This MAC register range have only a single address 0xF0 00.
All reader commands must send to this address to start the execution in MAC firmware.
Commands for Set frequency, change power or Tag Read / Write cycles are stored in this register.
All setup parameters for a reader function must defined in MAC registers, before the function can started.
If the command in the register, the reader starts imedially and respond with Start-, Stop- and Tag Access packets.

The reader response are depend from command type and setup in relevant MAC registers. Setup functions respond with Start- and Stop packets.
Inventory and Tag Access functions respond with Inventory packet type 0x0005 and Tag Access packet type 0x0006, when a functional tag is in antenna field.

The detail description for reader commands are in Impinj document “MAC Command Definitions”.
Please look into this document when you need more information’s.

# 5 Configuration of INDY R2k Reader

The reader setup have a lot of parameters and it is not easy to handle that without software assistance. We use for that our „MB Reader Tool“ and built macros with command sequences to make it easier and transparent for operators.
This tool monitor a copy of the command dialog and analyse data for operator.
In the next chapters below exist examples for hardware dialog with comments. Here you can find samples for complete reader setups and Read / Write sequences for Tag programming.

All parameters and setups start with Write commands to MAC register addresses. After MAC registers are set with data's, reader commands follow and start the reader function.
Write commands to MAC registers start with 0x01 and Read commands begin with 0x00.

Example Write MAC Register 0x0114 (RF Power for test port)

**01** 00 **14 01** 96 00 00 00

| | |
|---|---|
| 01 | Write to Mac Firmware MAC Firmware |
| 00 | Flag |
| 01 14 | MAC Register address 0x0114 |
| 96 | hex Wert for RF power in 1/10 dBm (15,0dBm) |

Example Read MAC Register 0x0114 (RF power test port)

```
00 00 14 01 00 00 00 00     Host command for Reader, Read MAC Register 0x0114
00 00 14 01 96 00 00 00     Reader responce
```

| | |
|---|---|
| 00 | Read MAC Firmware |
| 00 | Flag |
| 01 14 | MAC Register address 0x0114 |
| 96 | hex value RF power in 1/10 dBm (15,0dBm) |

## 5.1 Maschine Setup for Indy 2k Readers

The text sample below show a complete init sequence for Indy 2k reader in a PL30 perso machine

```
;##################################################################
;# Reader Initialization commands for ALL readers
;##################################################################
; Setup_MainConfig
; RF_power_off !!!
01 00 00 F0 18 00 00 00
```

```
; Ant. port 0 set, 100ms antenna slot active, count off
01 00 00 07 01 00 00 00
01 00 01 07 00 00 00 00
01 00 02 07 01 00 00 00
01 00 04 07 00 00 00 00
01 00 05 07 64 00 00 00
01 00 07 07 00 00 00 00

;Setup Inventory, EPC, Stop after 1. Tag
01 00 03 02 00 00 00 00
01 00 00 09 C0 00 00 00
01 00 01 09 40 00 00 00
01 00 02 09 00 00 00 00
01 00 03 09 F4 40 00 00
01 00 04 09 00 00 00 00
01 00 05 09 01 00 00 00

; Setup_TagAccessRead_EPC_96Bit, no passwords, no lock mask
01 00 01 0A 06 00 00 00
01 00 02 0A 01 00 00 00
01 00 03 0A 02 00 00 00
01 00 04 0A 06 00 00 00
01 00 05 0A 00 00 00 00
01 00 06 0A 00 00 00 00
01 00 07 0A 00 00 00 00
01 00 08 0A 00 00 00 00
00 00 05 00 00 00 00 00


;#################################################################
,# Reader Initialisation specific commands
;#################################################################
[COM1]
; RF_power_off !!!
01 00 00 F0 18 00 00 00
; set RF power 15,0dBm
01 00 14 01 96 00 00 00
01 00 06 07 96 00 00 00
00 00 05 00 00 00 00 00

; frequency set to 902,1MHz = 902100 = 0DC3D4
01 00 0A 01 D4 C3 0D 00
01 00 08 01 01 01 00 00
01 00 00 F0 27 00 00 00
```

```
[COM2]
; RF_power_off !!!
01 00 00 F0 18 00 00 00

; set test power 15,0dBm
01 00 14 01 96 00 00 00
01 00 06 07 96 00 00 00
00 00 05 00 00 00 00 00

; Working frequency set to 907,3MHz = 907300 = 0DD824
01 00 0A 01 24 D8 0D 00
01 00 08 01 01 01 00 00
01 00 00 F0 27 00 00 0
```

## 5.2 Command-Dialog between Maschine <==> Indy Reader:

Connect request, (only for USB connection required!)

```
---------------------------------------------
02:40:02.213 HostPC >> C0 06 00 00 00 00 00 00          ; Connect
- - - - - - - - - - - - - - - - - -
02:40:02.541 Reader << 24 03 49 00 6D 00 70 00          ; Impinj serial Num 01, code as hex
02:40:02.541 Reader << 69 00 6E 00 6A 00 53 00
02:40:02.541 Reader << 65 00 72 00 69 00 61 00
02:40:02.541 Reader << 6C 00 4E 00 75 00 6D 00
02:40:02.541 Reader << 30 00 31 00
---------------------------------------------
```

Antennenport setup (100ms on time, Port 0)

```
---------------------------------------------
02:40:21.430 HostPC >> 01 00 00 F0 18 00 00 00          ; RF power off
02:40:21.430 HostPC >> 01 00 00 07 01 00 00 00          ; Antenna cycles 01
02:40:21.430 HostPC >> 01 00 01 07 00 00 00 00          ; log antenna port 0
02:40:21.430 HostPC >> 01 00 02 07 01 00 00 00          ; antenna port active
02:40:21.430 HostPC >> 01 00 04 07 00 00 00 00          ; port for TX / RX set to 0
02:40:21.430 HostPC >> 01 00 05 07 64 00 00 00          ; antenna port 100ms available
02:40:21.430 HostPC >> 01 00 07 07 00 00 00 00          ; Inventory counter switch off
02:40:21.430 HostPC >> 00 00 05 00 00 00 00 00          ; check error?
- - - - - - - - - - - - - - - - - -
02:40:21.789 Reader << 01 00 00 00 02 00 00 00          ; answer start packet
02:40:21.805 Reader << 18 00 00 00 37 29 41 00
02:40:21.805 Reader << 01 00 01 00 02 00 00 00          ; answer stop packet
02:40:21.805 Reader << 37 29 41 00 00 00 00 00
02:40:21.805 Reader << 00 00 05 00 00 00 00 00          ; answer current error state
---------------------------------------------
```

RF power 15,0dBm (Test port + Tag Access)

```
---------------------------------------------
02:40:41.912 HostPC >> 01 00 00 F0 18 00 00 00          ; rf power off
02:40:41.912 HostPC >> 01 00 14 01 96 00 00 00          ; power antenna port 15,0dBm
02:40:41.912 HostPC >> 01 00 06 07 96 00 00 00          ; rf power test port 15,0dBm
02:40:41.912 HostPC >> 00 00 05 00 00 00 00 00          ; check error?
- - - - - - - - - - - - - - - - -
02:40:42.271 Reader << 01 00 00 00 02 00 00 00          ; answer start packet
02:40:42.271 Reader << 18 00 00 00 40 79 41 00
02:40:42.271 Reader << 01 00 01 00 02 00 00 00          ; answer stop packet
02:40:42.271 Reader << 40 79 41 00 00 00 00 00
02:40:42.271 Reader << 00 00 05 00 00 00 00 00          ; answer current error state
---------------------------------------------
```

Frequency 915,5MHz for Tag Access functions

```
---------------------------------------------
02:41:02.707 HostPC >> 01 00 00 F0 18 00 00 00          ; rf power off
02:41:02.707 HostPC >> 01 00 0A 01 2C F8 0D 00          ; frequency EtSi starts 1D, FCC with 0D
02:41:02.707 HostPC >> 01 00 08 01 01 01 00 00          ; set frequency CFG active
02:41:02.707 HostPC >> 01 00 00 F0 27 00 00 00          ; set frequency
- - - - - - - - - - - - - - - - -
02:41:03.285 Reader << 01 00 00 00 02 00 00 00          ; start packet (RF off)
02:41:03.285 Reader << 18 00 00 00 7B CA 41 00
02:41:03.285 Reader << 01 00 01 00 02 00 00 00          ; stop packet (RF off)
02:41:03.285 Reader << 7B CA 41 00 00 00 00 00
02:41:03.285 Reader << 01 00 00 00 02 00 00 00          ; start packet (Set frequency)
02:41:03.285 Reader << 27 00 00 00 80 CA 41 00
02:41:03.285 Reader << 01 00 01 00 02 00 00 00          ; stop packet (Set frequency)
02:41:03.285 Reader << 80 CA 41 00 00 00 00 00
---------------------------------------------
```

Inventory setup EPC 96Bit

```
---------------------------------------------
02:41:39.110 HostPC >> 01 00 03 02 00 00 00 00          ; Impinj Extensions off (FastID aus)
02:41:39.110 HostPC >> 00 00 06 00 00 00 00 00          ; check last error
02:41:39.110 HostPC >> 01 00 00 09 C0 00 00 00
02:41:39.110 HostPC >> 01 00 01 09 40 00 00 00          ; stop after 1. tag
02:41:39.110 HostPC >> 01 00 02 09 00 00 00 00
02:41:39.110 HostPC >> 01 00 03 09 F4 40 00 00
02:41:39.110 HostPC >> 01 00 05 09 01 00 00 00
02:41:39.110 HostPC >> 00 00 05 00 00 00 00 00          ; error check state
- - - - - - - - - - - - - - - - -
02:41:39.266 Reader << 00 00 06 00 00 00 00 00          ; answer last error
02:41:39.266 Reader << 00 00 05 00 00 00 00 00          ; answer current error state
---------------------------------------------
```

Tag Access Setup: Read EPC 96Bit

```
---------------------------------------------
02:42:36.714 HostPC >> 01 00 01 0A 06 00 00 00        ; retries counter set to 3, Bit 0 is not used!
02:42:36.714 HostPC >> 01 00 02 0A 01 00 00 00        ; memory bank 01 = EPC
02:42:36.714 HostPC >> 01 00 03 0A 02 00 00 00        ; offset counter 02 blocks
02:42:36.714 HostPC >> 01 00 04 0A 06 00 00 00        ; word counter 06 blocks
02:42:36.714 HostPC >> 01 00 05 0A 00 00 00 00        ; lock cfg register ==> no lock
02:42:36.714 HostPC >> 01 00 06 0A 00 00 00 00        ; acces PWD ==> 00 00 00 00
02:42:36.714 HostPC >> 01 00 07 0A 00 00 00 00        ; kill code ==> 00 00 00 00
02:42:36.714 HostPC >> 01 00 08 0A 00 00 00 00        ; config Tag write dat register ==> max. 128Bit
02:42:36.714 HostPC >> 00 00 05 00 00 00 00 00        ; check error?
- - - - - - - - - - - - - - - - -
02:42:36.808 Reader << 00 00 05 00 00 00 00 00        ; answer error state
---------------------------------------------
```

Start Inventory 96 Bit

```
---------------------------------------------
02:43:09.963 HostPC >> 01 00 00 F0 0F 00 00 00        ; inventory command = 0F
- - - - - - - - - - - - - - - - -
02:43:10.572 Reader << 01 00 00 00 02 00 00 00        ; start packet 00 00
02:43:10.572 Reader << 0F 00 00 00 B0 BB 43 00
02:43:10.572 Reader << 01 02 05 00 07 00 00 00        ;inventory packet
02:43:10.572 Reader << C7 BB 43 00 61 7D 04 01        ; PC Parameter 30 00 = 96Bit
02:43:10.572 Reader << 91 FE 00 00 30 00 AA AA        ; EPC: AA AA BB BB CC CC DD DD EE EE FF FF
02:43:10.572 Reader << BB BB CC CC DD DD EE EE        ; 4A 5B = CRC
02:43:10.588 Reader << FF FF 4A 5B 01 00 01 00        ; stop packet 00 01
02:43:10.588 Reader << 02 00 00 00 C7 BB 43 00
02:43:10.588 Reader << 00 00 00 00                    ; Mac Error Status 00 00 00 00 = no Error
---------------------------------------------
```

Start Tag Access Read 96Bit EPC

```
---------------------------------------------
02:43:33.087 HostPC >> 01 00 00 F0 10 00 00 00        ; start Tag Access comand 10
- - - - - - - - - - - - - - - - -
02:43:33.977 Reader << 01 00 00 00 02 00 00 00        ; start packet 00 00
02:43:33.977 Reader << 10 00 00 00 F9 15 44 00
02:43:33.977 Reader << 01 02 05 00 07 00 00 00        ; inventory packet
02:43:33.977 Reader << 0F 16 44 00 62 7D 04 01        ; PC Parameter 30 00 = 96Bit
02:43:33.977 Reader << 9A FE 00 00 30 00 AA AA        ; EPC: AA AA BB BB CC CC DD DD EE EE FF FF
02:43:33.977 Reader << BB BB CC CC DD DD EE EE        ; 4A 5B = CRC
02:43:33.977 Reader << FF FF 4A 5B 01 00 06 00        ; Tag Access packet 00 06
02:43:33.977 Reader << 06 00 00 00 14 16 44 00
02:43:33.977 Reader << C2 00 00 00 00 00 00 00        ; C2 =Read, 00 00 00 = no Error
02:43:33.977 Reader << AA AA BB BB CC CC DD DD        ; Chip data: AA AA BB BB CC CC DD DD EE EE FF FF
02:43:33.977 Reader << EE EE FF FF 01 00 01 00        ; stop packet 00 01
02:43:33.977 Reader << 02 00 00 00 14 16 44 00
02:43:33.977 Reader << 00 00 00 00                    ; Mac Error Status 00 00 00 00 = no Error
---------------------------------------------
```

Write Setup EPC 96 Bit

```
---------------------------------------------
02:44:00.476 HostPC >> 01 00 05 07 64 00 00 00          ; antenna port for 100ms available
02:44:00.476 HostPC >> 01 00 01 09 01 00 00 00          ; inventory counter set to 1, stop after first
                                                        founded tag
02:44:00.476 HostPC >> 01 00 11 09 00 00 00 00          ; EPC match disable
02:44:00.476 HostPC >> 01 00 01 0A 06 00 00 00          ; retries counter set to 03, Bit 0 are ignored!
02:44:00.476 HostPC >> 01 00 02 0A 01 00 00 00          ; EPC memory bank 01
02:44:00.476 HostPC >> 01 00 03 0A 02 00 00 00          ; offset counter 02 blocks
02:44:00.476 HostPC >> 01 00 04 0A 06 00 00 00          ; word counter 06 blocks
02:44:00.476 HostPC >> 01 00 06 0A 00 00 00 00          ; Access PWD 00 00 00 00
02:44:00.476 HostPC >> 01 00 07 0A 00 00 00 00          ; Kill code 00 00 00 00
02:44:00.476 HostPC >> 00 00 05 00 00 00 00 00          ; check error?
- - - - - - - - - - - - - - - - -
02:44:00.570 Reader << 00 00 05 00 00 00 00 00          ; answer error state
---------------------------------------------
```

Write new EPC data into Reader 96Bit, **No blockwrite**!

```
---------------------------------------------
02:44:31.193 HostPC >> 01 00 09 0A 34 12 00 00          ; write datablock 0      12 34
02:44:31.193 HostPC >> 01 00 0A 0A 78 56 01 00          ; write datablock 1      56 78
02:44:31.193 HostPC >> 01 00 0B 0A BC 9A 02 00          ; write datablock 2      9A BC
02:44:31.193 HostPC >> 01 00 0C 0A 0F DE 03 00          ; write datablock 3      DE 0F
02:44:31.193 HostPC >> 01 00 0D 0A 34 12 04 00          ; write datablock 4      12 34
02:44:31.193 HostPC >> 01 00 0E 0A 78 56 05 00          ; write datablock 5      56 78
02:44:31.193 HostPC >> 01 00 06 0A 00 00 00 00          ; write Access PWD: 00 00 00 00
02:44:31.193 HostPC >> 00 00 05 00 00 00 00 00          ; check error state
- - - - - - - - - - - - - - - - -
02:44:31.287 Reader << 00 00 05 00 00 00 00 00
---------------------------------------------
```

Write command (Tag Access write 96Bit, no blockwrite)

```
---------------------------------------------
02:45:22.191 HostPC >> 01 00 00 F0 11 00 00 00          ; tag write command
- - - - - - - - - - - - - - - - -
02:45:22.972 Reader << 01 00 00 00 02 00 00 00          ; start packet
02:45:22.972 Reader << 11 00 00 00 3B C0 45 00
02:45:22.972 Reader << 01 02 05 00 07 00 00 00          ; inventory packet, diagnostics
02:45:22.972 Reader << 59 C0 45 00 61 7D 04 01
02:45:22.972 Reader << 91 FE 00 00 30 00 AA AA          ; EPC: AA AA BB BB CC CC DD DD EE EE FF FF
02:45:22.988 Reader << BB BB CC CC DD DD EE EE
02:45:22.988 Reader << FF FF 4A 5B 01 00 06 00          ; Access packet, error code 00 00 00
02:45:22.988 Reader << 03 00 00 00 DB C0 45 00
02:45:22.988 Reader << C3 00 00 00 06 00 00 00          ; C3 =Write, 00 00 00 = no Error, 06 = 6 Blocks
02:45:22.988 Reader << 01 00 01 00 02 00 00 00          ; stop packet
02:45:22.988 Reader << E4 C0 45 00 00 00 00 00
---------------------------------------------
```

Setup Tag Acces Read EPC 96 Bit

```
---------------------------------------------
02:46:41.469 HostPC >> 01 00 01 0A 06 00 00 00          ; retries counter ==> 03, bit 0 reserved
02:46:41.469 HostPC >> 01 00 02 0A 01 00 00 00          ; memory bank 01 ==> EPC
02:46:41.469 HostPC >> 01 00 03 0A 02 00 00 00          ; offset block counter 02
02:46:41.469 HostPC >> 01 00 04 0A 06 00 00 00          ; word counter 06
02:46:41.469 HostPC >> 01 00 05 0A 00 00 00 00          ; lock configuartion, off
02:46:41.469 HostPC >> 01 00 06 0A 00 00 00 00          ; Access PWD: 00 00 00 00
02:46:41.469 HostPC >> 01 00 07 0A 00 00 00 00          ; Kill code: 00 00 00 00
02:46:41.469 HostPC >> 01 00 08 0A 00 00 00 00          ; Config Tag Write dat register, max 128Bits
02:46:41.469 HostPC >> 00 00 05 00 00 00 00 00          ; check error?
- - - - - - - - - - - - - - - - -
02:46:41.563 Reader << 00 00 05 00 00 00 00 00          ; answer error state
---------------------------------------------
```

Write “Access Code”: 87 65 43 21 to Reserd Memory Bank, **no block write**!

```
---------------------------------------------
02:14:17.058 HostPC >> 01 00 02 0A 00 00 00 00          ; memory bank 00 ==> Resered bank
02:14:17.058 HostPC >> 01 00 03 0A 02 00 00 00          ; offset counter 02, memory map: ACCESS
02:14:17.058 HostPC >> 01 00 04 0A 02 00 00 00          ; Block counter 02 (32 Bits)
02:14:17.058 HostPC >> 01 00 08 0A 00 00 00 00          ; Config Tag Write dat register,max. 128Bits
02:14:17.058 HostPC >> 01 00 09 0A 65 87 00 00          ; ACCESS code datablock 0, 87 65
02:14:17.058 HostPC >> 01 00 0A 0A 21 43 01 00          ; Access code datablock 1, 43 21
02:14:17.058 HostPC >> 01 00 06 0A 00 00 00 00          ; Access PWD ==> 00 00 00 00
02:14:17.058 HostPC >> 01 00 00 F0 11 00 00 00          ; write comand (no block write)
- - - - - - - - - - - - - - - - -
02:14:17.871 Reader << 01 00 00 00 02 00 00 00          ; start packet
02:14:17.871 Reader << 11 00 00 00 CC 6D 0F 00
02:14:17.871 Reader << 01 02 05 00 07 00 00 00          ; inventory packet + diagnostics
02:14:17.871 Reader << E4 6D 0F 00 6A 94 04 01
02:14:17.871 Reader << 00 00 00 00 30 00 30 08          ; EPC: 30 08 33 B2 DD D0 01 40 00 00 00 00
02:14:17.871 Reader << 33 B2 DD D9 01 40 00 00
02:14:17.871 Reader << 00 00 39 BB 01 00 06 00          ; Tag Access packet
02:14:17.871 Reader << 03 00 00 00 F6 6D 0F 00
02:14:17.871 Reader << C3 00 00 00 02 00 00 00          ; C3 = Write, 00 00 00 = no Error, 02 Blocks
02:14:17.886 Reader << 01 00 01 00 02 00 00 00          ; stop packet
02:14:17.886 Reader << F7 6D 0F 00 00 00 00 00
---------------------------------------------
```

Write “Kill Code”: AB CD EF 01 in Reserd Memory Bank, **no block write!**

```
---------------------------------------------
02:16:07.484 HostPC >> 01 00 02 0A 00 00 00 00        ; memory bank 00 ==> Resered bank
02:16:07.484 HostPC >> 01 00 03 0A 00 00 00 00        ; offset counter 00, memory map: Kill data
02:16:07.484 HostPC >> 01 00 04 0A 02 00 00 00        ; Block counter 02
02:16:07.484 HostPC >> 01 00 08 0A 00 00 00 00        ; Config Tag Write dat register, max. 128 bits
02:16:07.484 HostPC >> 01 00 09 0A CD AB 00 00        ; KILL code datablock 0, AB CD
02:16:07.484 HostPC >> 01 00 0A 0A 01 EF 01 00        ; KILL code datablock 1, EF 01
02:16:07.484 HostPC >> 01 00 06 0A 00 00 00 00        ; ACCESS PWD ==> 00 00 00 00
02:16:07.484 HostPC >> 01 00 00 F0 11 00 00 00        ; Tag ACCESS Write comand
- - - - - - - - - - - - - - - -
02:16:08.281 Reader << 01 00 00 00 02 00 00 00        ; start packet
02:16:08.281 Reader << 11 00 00 00 24 1D 11 00
02:16:08.281 Reader << 01 02 05 00 07 00 00 00        ; Inventory packet
02:16:08.281 Reader << 34 1D 11 00 6A 94 04 01
02:16:08.281 Reader << 00 00 00 00 30 00 30 08        ; EPC: 30 08 330B2 DD D9 01 40 00 00 00 00
02:16:08.296 Reader << 33 B2 DD D9 01 40 00 00
02:16:08.296 Reader << 00 00 39 BB 01 00 06 00        ; Tag Access packet
02:16:08.296 Reader << 03 00 00 00 46 1D 11 00
02:16:08.296 Reader << C3 00 00 00 02 00 00 00        ; C3 = Write, 00 00 00 = no Error, 02 Blocks
02:16:08.296 Reader << 01 00 01 00 02 00 00 00        ; stop packet
02:16:08.296 Reader << 47 1D 11 00 00 00 00 00
---------------------------------------------
```

Read “Access Code”,  32 Bit

```
---------------------------------------------
02:16:47.735 HostPC >> 01 00 02 0A 00 00 00 00        ; memory bank 00 ==> RESERVED
02:16:47.735 HostPC >> 01 00 03 0A 02 00 00 00        ; offset counter 02 , skip Kill code, 2 blocks
02:16:47.735 HostPC >> 01 00 04 0A 02 00 00 00        ; Block counter 02, 32 Bits
02:16:47.735 HostPC >> 01 00 00 F0 10 00 00 00        ; Tag access read comand
- - - - - - - - - - - - - - - -
02:16:48.563 Reader << 01 00 00 00 02 00 00 00        ; start packet
02:16:48.563 Reader << 10 00 00 00 59 BA 11 00
02:16:48.563 Reader << 01 02 05 00 07 00 00 00        ; inventory packet + diagnostics
02:16:48.563 Reader << 6B BA 11 00 6A 94 04 01
02:16:48.563 Reader << 00 00 00 00 30 00 30 08        ; EPC: 30 08 33 B2 DD D9 01 40 00 00 00 00
02:16:48.563 Reader << 33 B2 DD D9 01 40 00 00        ; 39 BB = CRC
02:16:48.563 Reader << 00 00 39 BB 01 00 06 00        ; Tag access packet
02:16:48.563 Reader << 04 00 00 00 6F BA 11 00
02:16:48.563 Reader << C2 00 00 00 00 00 00 00        ; C2 = Read, 00 00 00 = no Error
02:16:48.563 Reader << 87 65 43 21 01 00 01 00        : Chip data: 87 65 43 21 (Access code)
02:16:48.563 Reader << 02 00 00 00 6F BA 11 00        ; stop packet
02:16:48.563 Reader << 00 00 00 00
---------------------------------------------
```

Read “Kill Code” from Reserved Bank

```
---------------------------------------------
02:17:08.470 HostPC >> 01 00 02 0A 00 00 00 00          ; memory bank 00 ==> Resered bank
02:17:08.470 HostPC >> 01 00 03 0A 00 00 00 00          ; offset counter 00 memory map: Access data
02:17:08.470 HostPC >> 01 00 04 0A 02 00 00 00          ; word counter 02
02:17:08.470 HostPC >> 01 00 00 F0 10 00 00 00          ; Tag access read comand
- - - - - - - - - - - - - - - - -
02:17:09.298 Reader << 01 00 00 00 02 00 00 00          ; start packet
02:17:09.298 Reader << 10 00 00 00 61 0B 12 00
02:17:09.298 Reader << 01 02 05 00 07 00 00 00          ; inventory packet + diagnostics
02:17:09.298 Reader << 71 0B 12 00 6A 94 04 01
02:17:09.298 Reader << 00 00 00 00 30 00 30 08
02:17:09.298 Reader << 33 B2 DD D9 01 40 00 00          ; EPC: 30 08 33 B2 DD D9 01 40 00 00 00 00
02:17:09.298 Reader << 00 00 39 BB 01 00 06 00          ; Tag access packet
02:17:09.298 Reader << 04 00 00 00 75 0B 12 00
02:17:09.298 Reader << C2 00 00 00 00 00 00 00
02:17:09.298 Reader << AB CD EF 01 01 00 01 00          ; Chip data: AB CD EF 01 (Kill code)
02:17:09.298 Reader << 02 00 00 00 75 0B 12 00          ; stop packet
02:17:09.314 Reader << 00 00 00 00
```

### With Blockwrite

Write new EPC data into Reader 96Bit, with Blockwrite!

Write command (Tag Access Write 96Bit, with Blockwrite)

Write Access Code: 87 65 43 21 to Reserd Memory Bank, with Blockwrite!

Write Kill Code: AB CD EF 01 in Reserd Memory Bank, with Blockwrite

Write PC Parameter in EPC Speicherbank

PC = 30 00 = 96Bit
PC = 40 00 = 128Bit

## 5.3 Sampes for memory bank access with EPC1 Gen 2 Chips

Command-dialog for Write three memory banks with Lock
Perma-Lock: EPC, KILL, ACCESS
Secure Lock: USER
Access Password: 87654321
Sample for Alien Higgs 3 chip

### 5.3.1 EPC Memory Bank

**==> Setup EPC 96Bit**
```
11:45:18.539 > von Maschine: 01 00 01 09 40 00 00 00
11:45:18.539 > von Maschine: 01 00 11 09 00 00 00 00
11:45:18.555 > von Maschine: 01 00 01 0A 06 00 00 00
11:45:18.555 > von Maschine: 01 00 02 0A 01 00 00 00
11:45:18.555 > von Maschine: 01 00 03 0A 02 00 00 00
11:45:18.555 > von Maschine: 01 00 04 0A 06 00 00 00
11:45:18.555 > von Maschine: 01 00 06 0A 00 00 00 00
11:45:18.555 > von Maschine: 01 00 07 0A 00 00 00 00
11:45:18.555 > von Maschine: 01 00 08 0A 00 00 00 00
11:45:18.555 > von Maschine: 00 00 05 00 00 00 00 00
11:45:18.555 < von Reader: 00 00 05 00 00 00 00 00
```

**==> Send new EPC Data  for Reader**
```
11:45:22.477 > von Maschine: 01 00 09 0A 11 11 00 00
11:45:22.477 > von Maschine: 01 00 0A 0A 11 11 01 00
11:45:22.477 > von Maschine: 01 00 0B 0A 11 11 02 00
11:45:22.477 > von Maschine: 01 00 0C 0A 11 11 03 00
11:45:22.477 > von Maschine: 01 00 0D 0A 11 11 04 00
11:45:22.477 > von Maschine: 01 00 0E 0A 11 11 05 00
11:45:22.477 > von Maschine: 01 00 06 0A 00 00 00 00
11:45:22.477 > von Maschine: 00 00 05 00 00 00 00 00
11:45:22.477 < von Reader: 00 00 05 00 00 00 00 00
```

**==> Write new EPC Data into Chip Memory**
```
11:45:23.352 > von Maschine: 01 00 00 F0 11 00 00 00
11:45:23.352 < von Reader: 01 00 00 00 02 00 00 00
11:45:23.368 < von Reader: 11 00 00 00 63 51 61 00
11:45:23.368 < von Reader: 01 12 05 00 07 00 00 00
11:45:23.368 < von Reader: 7D 51 61 00 5F 7F 04 01
11:45:23.368 < von Reader: 81 FE 5F 3F 30 00 E2 00
11:45:23.368 < von Reader: 83 66 66 05 01 38 26 90
11:45:23.368 < von Reader: 08 91 A8 38 01 00 06 00
11:45:23.368 < von Reader: 03 00 00 00 B7 51 61 00
11:45:23.368 < von Reader: C3 00 00 00 06 00 00 00
11:45:23.368 < von Reader: 01 00 01 00 02 00 00 00
11:45:23.368 < von Reader: B8 51 61 00 00 00 00 00
```

### 5.3.2 USER Memory Bank

**==> Reader Setup User Memory**
```
11:45:46.493 > von Maschine: 01 00 02 0A 03 00 00 00
11:45:46.493 > von Maschine: 01 00 03 0A 00 00 00 00
11:45:46.493 > von Maschine: 01 00 04 0A 03 00 00 00
11:45:46.493 > von Maschine: 01 00 08 0A 00 00 00 00
11:45:46.493 > von Maschine: 00 00 05 00 00 00 00 00
11:45:46.493 < von Reader: 00 00 05 00 00 00 00 00
```

**==> Send User Memory Data for Reader (kein Blockwrite)**

```
11:45:47.165 > von Maschine: 01 00 09 0A BB AA 00 00
11:45:47.165 > von Maschine: 01 00 0A 0A DD CC 01 00
11:45:47.165 > von Maschine: 01 00 0B 0A FF EE 02 00
11:45:47.165 > von Maschine: 00 00 05 00 00 00 00 00
11:45:47.165 < von Reader: 00 00 05 00 00 00 00 00
```

**==> Write USER Data into Memory**

```
11:45:47.993 > von Maschine: 01 00 00 F0 11 00 00 00
11:45:47.993 < von Reader: 01 00 00 00 02 00 00 00
11:45:47.993 < von Reader: 11 00 00 00 FB B1 61 00
11:45:47.993 < von Reader: 01 12 05 00 07 00 00 00
11:45:47.993 < von Reader: 13 B2 61 00 5D 80 04 01
11:45:47.993 < von Reader: 74 FE 5D 3F 30 00 11 11
11:45:47.993 < von Reader: 11 11 11 11 11 11 11 11
11:45:47.993 < von Reader: 11 11 D8 B2 01 00 06 00
11:45:47.993 < von Reader: 03 00 00 00 30 B2 61 00
11:45:47.993 < von Reader: C3 00 00 00 03 00 00 00
11:45:47.993 < von Reader: 01 00 01 00 02 00 00 00
11:45:47.993 < von Reader: 30 B2 61 00 00 00 00 00
```

### 5.3.3 Reserved Memory Bank

**==> Reader Setup for Access Code Write**

```
11:45:37.883 > von Maschine: 01 00 01 09 40 00 00 00
11:45:37.883 > von Maschine: 01 00 11 09 00 00 00 00
11:45:37.883 > von Maschine: 01 00 01 0A 06 00 00 00
11:45:37.883 > von Maschine: 01 00 02 0A 00 00 00 00
11:45:37.883 > von Maschine: 01 00 03 0A 02 00 00 00
11:45:37.883 > von Maschine: 01 00 04 0A 02 00 00 00
11:45:37.883 > von Maschine: 01 00 06 0A 00 00 00 00
11:45:37.883 > von Maschine: 01 00 07 0A 00 00 00 00
11:45:37.883 > von Maschine: 01 00 08 0A 00 00 00 00
11:45:37.883 > von Maschine: 00 00 05 00 00 00 00 00
11:45:37.883 < von Reader: 00 00 05 00 00 00 00 00
```

**==> Send Access Code data to reader for Write**

```
11:45:39.540 > von Maschine: 01 00 09 0A 65 87 00 00
11:45:39.540 > von Maschine: 01 00 0A 0A 21 43 01 00
11:45:39.540 > von Maschine: 00 00 05 00 00 00 00 00
11:45:39.540 < von Reader: 00 00 05 00 00 00 00 00
```

**==> Write Access Code into Chip memory**

```
11:45:41.962 > von Maschine: 01 00 00 F0 11 00 00 00
11:45:41.962 < von Reader: 01 00 00 00 02 00 00 00
11:45:41.962 < von Reader: 11 00 00 00 9B 9A 61 00
11:45:41.962 < von Reader: 01 12 05 00 07 00 00 00
11:45:41.962 < von Reader: AD 9A 61 00 5D 7E 04 01
11:45:41.962 < von Reader: 74 FE 1D 3F 30 00 11 11
11:45:41.962 < von Reader: 11 11 11 11 11 11 11 11
11:45:41.962 < von Reader: 11 11 D8 B2 01 00 06 00
11:45:41.993 < von Reader: 03 00 00 00 C0 9A 61 00
11:45:41.993 < von Reader: C3 00 00 00 02 00 00 00
11:45:41.993 < von Reader: 01 00 01 00 02 00 00 00
11:45:41.993 < von Reader: C0 9A 61 00 00 00 00 00
```

**Set Access Passwort in MAC Register for Reader <=> Chip Communication**

```
11:46:09.275 > von Maschine: 01 00 06 0A 21 43 65 87                    ; Access Code PWD 87 65 43 21
11:46:09.275 > von Maschine: 00 00 05 00 00 00 00 00
11:46:09.275 < von Reader: 00 00 05 00 00 00 00 00
```

## 5.4 Samples for Lock Functions with EPC 1 Gen2 Chips

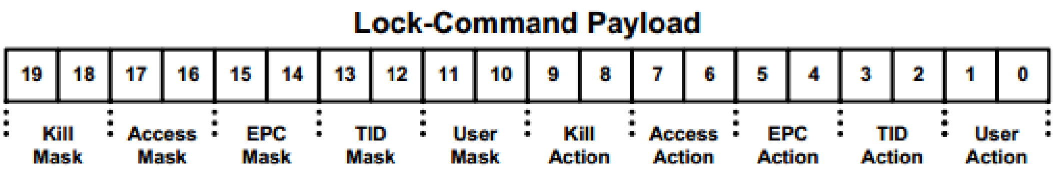
Before you can Lock a Tag, the MAC register 0A 08h (Tag Write Selector) must set to value 00 00 00 00h. We do this as default with Init sequence for the reader.
If the register not set to zero, you can get Error codes when you start Lock functions, e.g. error code 00 09

**11:46:06.024 > von Maschine: 01 00 08 0A 00 00 00 00 ; Write Tag Selector**

**Set Lock Mask for Perma Lock EPC, Kill and Access and PWD Lock for User (Neology)**

```
11:46:06.024 > von Maschine: 01 00 05 0A F2 CF 0F 00          ;Lock Mask
11:46:06.024 > von Maschine: 00 00 05 00 00 00 00 00
11:46:06.024 < von Reader: 00 00 05 00 00 00 00 00
```

| | | | |
|---|---|---|---|
| TAGACC LOCK CFG | 0A05 | 01 00 05 0A 00 00 00 00<br>00 00 05 00 00 00 00 00 | Lock action & mask Register gemäss ISO18k6C Spec<br>Bit 0:9 action Bits, 19:10mask, 31:20 reserve |
| | | 01 00 05 0A 00 00 00 00 | no change |
| | | 01 00 05 0A 00 00 03 00 | ACCESS Accessible |
| | | 01 00 05 0A 40 00 03 00 | ACCESS perma unlock |
| | | 01 00 05 0A 80 00 03 00 | ACCESS secured accessible |
| | | 01 00 05 0A C0 00 03 00 | ACCESS always not accessible (permalock) |
| | | 01 00 05 0A 00 01 0C 00 | Kill perma unlock |
| | | 01 00 05 0A 00 02 0C 00 | Kill secured accessible |
| | | 01 00 05 0A 00 03 0C 00 | Kill allways not accessible (permalock) |
| | | 01 00 05 0A 10 C0 00 00 | EPC perma unlock |
| | | 01 00 05 0A 20 C0 00 00 | EPC Secured writeable |
| | | 01 00 05 0A 30 C0 00 00 | EPC not writeable (permalock) |
| | | 01 00 05 0A 04 30 00 00 | TID perma unlock |
| | | 01 00 05 0A 08 30 00 00 | TID secured writeable |
| | | 01 00 05 0A 0C 30 00 00 | TID always not writeable (permalock) |
| | | 01 00 05 0A 01 0C 00 00 | USER perma unlock |
| | | 01 00 05 0A 02 0C 00 00 | USER secured writeable |
| | | 01 00 05 0A 03 0C 00 00 | USER always not writeable |
| | | 01 00 05 0A A0 C0 03 00 | Secured EPC, ACCESS |
| | | 01 00 05 0A A0 C2 0F 00 | secured EPC, ACCESS, KILL |
| | | 01 00 05 0A 30 C3 0C 00 | Perma Lock: Kill, EPC |
| | | 01 00 05 0A F0 C3 0F 00 | Perma Lock: EPC, ACCESS, Kill |
| | | 01 00 05 0A F3 CF 0F 00 | Perma Lock: EPC, User, Kill, Access |
| | | **01 00 05 0A FF FF 0F 00** | **Impinj Monza R6 Perma Lock all, Access + Kill is not readable and can not use** |



**Masks and Associated Action Fields**

| | Kill pwd | | Access pwd | | EPC memory | | TID memory | | User memory | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |
| *Mask* | skip/ write | skip/ write | skip/ write | skip/ write | skip/ write | skip/ write | skip/ write | skip/ write | skip/ write | skip/ write |
| | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| *Action* | pwd read/ write | perma lock | pwd read/ write | perma lock | pwd write | perma lock | pwd write | perma lock | pwd write | perma lock |

**Figure 6.24 – *Lock* payload and usage**

**Table 6.43 – *Lock* Action-field functionality**

| pwd-write | permalock | Description |
|---|---|---|
| 0 | 0 | Associated memory bank is writeable from either the **open** or **secured** states. |
| 0 | 1 | Associated memory bank is permanently writeable from either the **open** or **secured** states and may never be locked. |
| 1 | 0 | Associated memory bank is writeable from the **secured** state but not from the **open** state. |
| 1 | 1 | Associated memory bank is not writeable from any state. |
| **pwd-read/write** | **permalock** | **Description** |
| 0 | 0 | Associated password location is readable and writeable from either the **open** or **secured** states. |
| 0 | 1 | Associated password location is permanently readable and writeable from either the **open** or **secured** states and may never be locked. |
| 1 | 0 | Associated password location is readable and writeable from the **secured** state but not from the **open** state. |
| 1 | 1 | Associated password location is not readable or writeable from any state. |

**==> Lock Comand, the type a Write protection and memeory banks will defined by Lock Mask**

```
11:46:12.665 > von Maschine: 01 00 00 F0 12 00 00 00
11:46:12.665 < von Reader: 01 00 00 00 02 00 00 00
11:46:12.665 < von Reader: 12 00 00 00 50 12 62 00
11:46:12.665 < von Reader: 01 12 05 00 07 00 00 00
11:46:12.665 < von Reader: 64 12 62 00 5E 7F 04 01
11:46:12.665 < von Reader: 7B FE 5D 3F 34 00 11 11
11:46:12.665 < von Reader: 11 11 11 11 11 11 11 11
11:46:12.665 < von Reader: 11 11 25 17 01 00 06 00
11:46:12.665 < von Reader: 03 00 00 00 76 12 62 00
11:46:12.665 < von Reader: C5 00 00 00 00 00 00 00
11:46:12.665 < von Reader: 01 00 01 00 02 00 00 00
11:46:12.665 < von Reader: 76 12 62 00 00 00 00 00
```

## 5.5 Example Command sequence

**TID Read, EPC Write, Access Code Write, Lock for EPC and Access Code**

**1. Setup TID 96 Bit**

```
HostPC >> 01 00 02 0A 02 00 00 00        ; memory bank 2 = TID
HostPC >> 01 00 03 0A 00 00 00 00        ; offset counter 0 blocks
HostPC >> 01 00 04 0A 06 00 00 00        ; word counter 6 blocks = 96 Bit
HostPC >> 01 00 00 F0 10 00 00 00        ; start Read comand
```

**2. Setup EPC 96 Bit**

```
HostPC >> 01 00 02 0A 01 00 00 00        ; memory bank 02 = TID
HostPC >> 01 00 03 0A 02 00 00 00        ; offset counter 2 blocks
HostPC >> 01 00 04 0A 06 00 00 00        ; word counter 6 blocks = 96 Bit
```

**3. Send new EPC Daten to Reader (0123456789ABCDEF01234567)**

```
HostPC >> 01 00 09 0A 23 01 00 00
HostPC >> 01 00 0A 0A 67 45 01 00
HostPC >> 01 00 0B 0A AB 89 02 00
HostPC >> 01 00 0C 0A EF CD 03 00
HostPC >> 01 00 0D 0A 23 01 04 00
HostPC >> 01 00 0E 0A 67 45 05 00
```

**4. Store new EPC Data in Chip memeory (Write)**

```
HostPC >> 01 00 00 F0 11 00 00 00        ; start Write command 11
```

**5. Setup for Access Code**

```
HostPC >> 01 00 02 0A 00 00 00 00        ; memory bank 00 = Reserved
HostPC >> 01 00 03 0A 02 00 00 00        ; offset counter 02 blocks
HostPC >> 01 00 04 0A 02 00 00 00        ; word counter 02 blocks = 32 Bit
```

**6. Send Access PWD to Reader (12345678)**

```
HostPC >> 01 00 09 0A 34 12 00 00
HostPC >> 01 00 0A 0A 78 56 01 00
```

**7. Write Access Code in Chip**

```
HostPC >> 01 00 00 F0 11 00 00 00        ; start Write command 11
```

**8. Set Lock MASK for Perma Lock EPC**

```
HostPC >> 01 00 05 0A 30 C0 00 00
```

ODER

**8. Set Lock MASK for Secure Lock EPC & Access**

```
HostPC >> 01 00 05 0A A0 C0 03 00
```

**9. Set Access PWD for Reader <=> Tag Communication (12345678)**

```
HostPC >> 01 00 06 0A 78 56 34 12        ; Access Code PWD 12345678
```

**10. Lock Command for all in der Lock Mask defined memeory banks**

```
HostPC >> 01 00 00 F0 12 00 00 00        ; Lock command 12
```

### 5.6 Example, how modify a Secure locked EPC bank with new data

If a memory bank Secure locked, you can modify the memory bank only when you use the Access password for that. The access password is stored in chip Reserved memory bank area.
Lock cycles before activate it as Write protection password and now is the bank locked.

***When a Reserved Memory bank have an Access password stored and one bank of the chip are locked with Secure Lock, then you can't read the Access password from Reserved memory bank. Here is a read protection active.***

When you want modify a Secure locked memory bank, then you must follow a special command sequence to modify locked data.

Sample to modify EPC when EPC Bank is locked with Access password:

1. Tag Access Setup set to EPC Bank
2. new EPC Daten send to Reader (store in MAC register 09 0Ah and follow)
3. Send active Access password to the reader, (MAC register 06 00h)
4. Start Write command for Tag Access to Write new EPC data with password into EPC bank

***The Access password must send to reader (MAC 06 00h) directly before you send the Write command. Between password information and Write command are no other actions allowed or you get the Error code memory bank is locked.( 0x040000)***

The Access password can't read out from reader. You must know it before you can start with the sequence.
If you want read the password MAC register 06 00h, you get as reader response FF FF FF FF.

# 6 HOST Paket Definition

All reader response packets comes with special packet types and code numbers.

The syntax of these packets are descripted in document „HOST Packet Definitions" from Impinj.
When the MAC firmware start a setup or test functions , then you get Start- and Stop- packets as frame around all other packet types.

For the following reader functions create the reader a "packet frame" with Start- and Stop-messages:
- Frequency setup
- RF power /ON/OFF
- Chip Inventory (0x0F)
- Chip Read (0x10)
- Chip Write (0x11) no Blockwrite mode
- Chip Write (0x1F) no Blockwrite mode
- Chip Lock (0x12)

Important packet types for our machines:

| | |
|---|---|
| 0000 | Start Packet |
| 0001 | End Packet |
| 0004 | Inventory beginn |
| 0005 | Inventory Packet (PC-Value, EPC, Time stamp, RSSI) |
| 0006 | Tag Access Packet (with Chip Data, Chip Error Codes, Test Error Codes) |
| 0009 | Inventory end |

# 7 Error Codes

## 7.1 MAC Error Codes

(Error request for actual status „0x0005“ or request last  Error Code  „0x0006“)

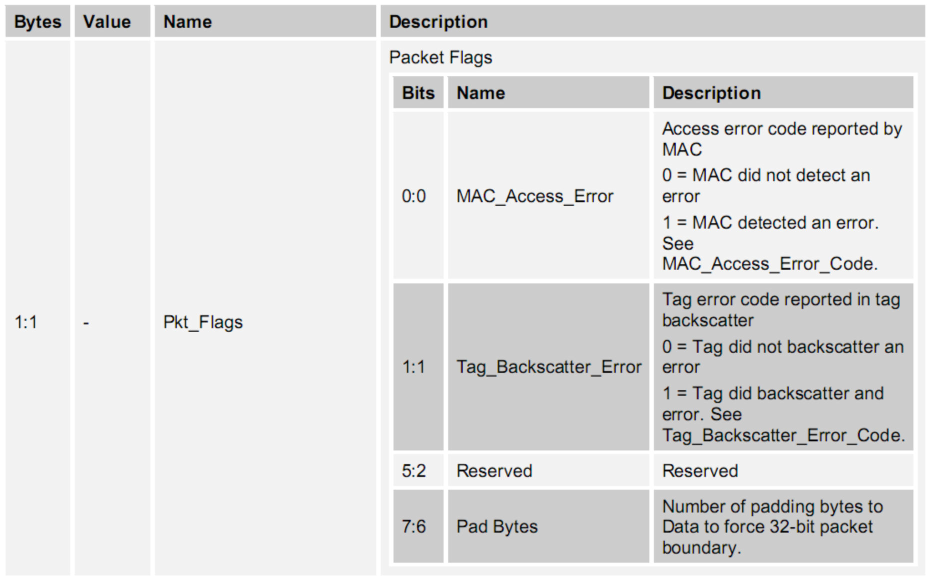
Example Error codes:

| | |
|---|---|
| 0000 | no Error |
| 0001 | invalid command |
| 002C | antenna port not available |
| 0113 | unknown / unsupported HOST command |
| 011A | write fault to Read-Only Mac register |
| 0300 | set wrong frequency channel |
| 0302 | PLL lock fault |
| 0307 | power amplifier to hot |
| 0308 | Temp difference between power amp and ambient to great |
| 0309 | antenna or coax cable parameters wrong, SMA connector fault |
| 0318 | real RF power higher as target value |
| 0330 | general health check failure |
| 0335 | PA bias calibration Error |
| 033C | general cross gain error |

A complete list of Error codes you can find in document “Host error code definitions”.

## 7.2 TAG Error Codes (Packet type 0006)

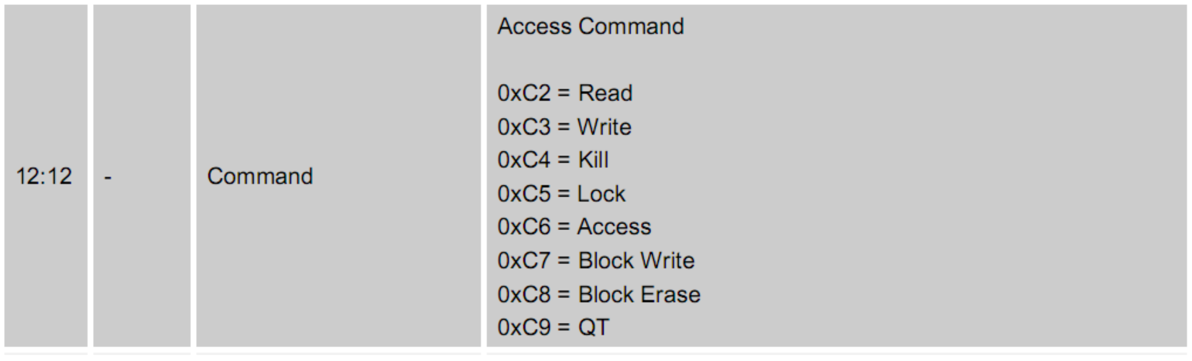
Tag Access functions respond with packet type 6.



| Bytes | Value | Name | Description |
|---|---|---|---|
| 1:1 | - | Pkt_Flags | Packet Flags |

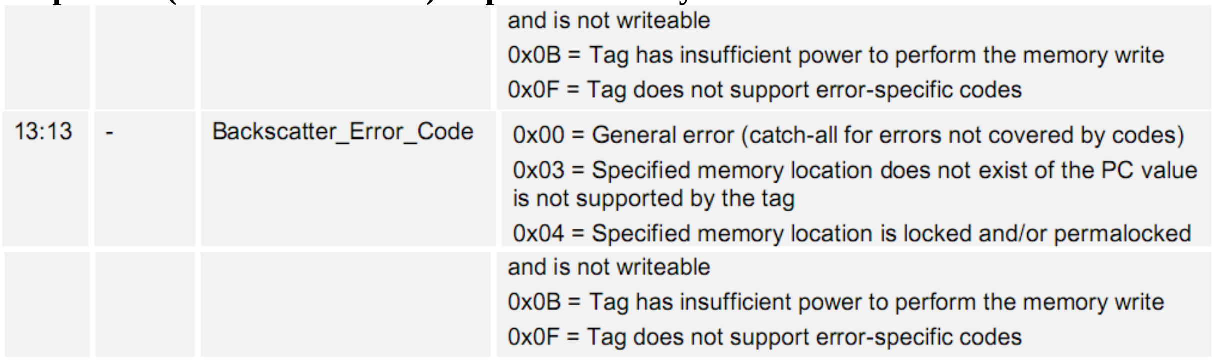
| Bits | Name | Description |
|---|---|---|
| 0:0 | MAC_Access_Error | Access error code reported by MAC<br>0 = MAC did not detect an error<br>1 = MAC detected an error. See MAC_Access_Error_Code. |
| 1:1 | Tag_Backscatter_Error | Tag error code reported in tag backscatter<br>0 = Tag did not backscatter an error<br>1 = Tag did backscatter and error. See Tag_Backscatter_Error_Code. |
| 5:2 | Reserved | Reserved |
| 7:6 | Pad Bytes | Number of padding bytes to Data to force 32-bit packet boundary. |

**Byte no. 12 proive a command code for the test running test function**



| | | | |
|---|---|---|---|
| 12:12 | - | Command | Access Command<br><br>0xC2 = Read<br>0xC3 = Write<br>0xC4 = Kill<br>0xC5 = Lock<br>0xC6 = Access<br>0xC7 = Block Write<br>0xC8 = Block Erase<br>0xC9 = QT |

**Chip Error (Backscatter Error) responce are in byte 13.**



| | | | |
|---|---|---|---|
| | | | and is not writeable<br>0x0B = Tag has insufficient power to perform the memory write<br>0x0F = Tag does not support error-specific codes |
| 13:13 | - | Backscatter_Error_Code | 0x00 = General error (catch-all for errors not covered by codes)<br>0x03 = Specified memory location does not exist of the PC value is not supported by the tag<br>0x04 = Specified memory location is locked and/or permalocked |
| | | | and is not writeable<br>0x0B = Tag has insufficient power to perform the memory write<br>0x0F = Tag does not support error-specific codes |

**Test Errors monitor the MAC firmware in byte 14 and 15.**

| | | | |
|---|---|---|---|
| 15:14 | - | Mac_Access_Error_Code | If the MAC detects an error (i.e. the MAC_Access_Error flag is set), and none of the error specific bits are set in the flags field, this field contains a 16-bit error code.<br><br>0x0000 = No error<br>0x0001 = Handle mismatch<br>0x0002 = CRC error on tag response<br>0x0003 = No tag reply<br>0x0004 = Invalid password<br>0x0005 = Zero kill password<br>0x0006 = Tag lost<br>0x0007 = Command format error<br>0x0008 = Read count invalid<br>0x0009 = Out of retries<br>0xFFFF = Operation failed |