OpenDev ODRFID(,-M,-N,-E), ODRFID-485, RS485-IO

Open Development LLC

2021-03-29

Contents

1 Overview

This reference manual provides operating instructions, command references for different firmwares, and other detailed product information. If you have any questions, please visit our Knowledge Base, and if you need further assistance, send us a Technical assistance request.

Open Development LLC https://open-dev.ru

2 Features

2.1 ODRFID

- Reader is a bus-powered USB 2.0 full-speed device with micro-USB connector. The device dimensions are 10.3x6.8x1.1cm.
- Indication: a green LED (optional) and a buzzer.
- Available firmware types include:
  - HID (emulates a keyboard) - out-of-the-box OS support. No driver installation on Microsoft Windows, Apple macOS and Linux;
  - CDC/ACM (virtual serial port) - works out of the box in Microsoft Windows 8 and 10, Apple macOS and Linux, a driver for Microsoft Windows 7 (32 bit and 64 bit) is available.
  - UART via RS485 (supports the same command set as the CDC variant, embedded in a binary serial protocol)
  - MODBUS over RS485 (supports the same command set as the CDC variant, embedded in a MODBUS RTU protocol)
- Optionally supports 125kHz EM41xx compatible tags and 13.56 MHz RFID MIFARE tags (Classic 1K/4K/Mini, Ultralight, NTAG, Plus)
- Extensible and future-proof design: the device firmware can be easily changed or updated using the built-in HID USB Bootloader.
- Hardware revisions of the device and their differences are described in the Revisions section

2.2 ODRFID-485

- Reader is powered by DC 5V-20V.
- Data interface is RS-485.
- Indication: LEDs (green and red) and a buzzer.
- Available firmware types include:
  - AT-commands over RS485 bus.
  - MODBUS-RTU protocol.
- Supports 13.56 MHz RFID MIFARE tags (Classic 1K/4K/Mini, Ultralight, NTAG, Plus)

2.3 RS485-IO

- The device is powered by DC 5V-20V.
- Data interface is RS-485.
- Indication: a green LED
- Available firmware types:
  - AT-commands over RS485 bus.
  - MODBUS-RTU protocol (coming soon)

3 ODRFID (CDC Firmware)

The device appears as a virtual serial port (COM port) by implementing the USB Communications Device Class, Abstract Control Model (CDC/ACM) specification. All major operating systems, including Windows, Linux, and macOs support such devices out-of-the-box.

Differences in the device revisions are described in the Hardware Revisions section.

An AT command set is provided to query tags, read and write data, and configure the device itself.

3.1 Port settings

The device represents itself as a virtual COM port. As such, port settings, like communication speed, parity or stop bit settings will be ignored. No specific configuration is necessary and the port can be operated at any settings (without influencing the communication speed).

3.2 General Frame Format

- Host-to-Device data must start with an `AT` keyword and end with a carriage-return character (ASCII code 13 decimal, 0x0d hexadecimal, *"\r"* as C string literal)
- Device-to-Host frames start and end with a carriage-return character followed by a line-feed character (ASCII codes 13,10 decimal, 0x0d,0x0a hexadecimal, *"\r\n"* as C string literal).
- Device-to-Host response to a Host-to-Device request consists of one or more frames, the last one being `\r\nOK\r\n` or `\r\nERROR\r\n`, depending on whether the request/command has been completed successfully or failed.
- Do **not** add any unnecessary white space characters (space, back-space, tab, carriage-return, line-feed, etc.) to the request data, they will **not** be removed by the command-line parser leading to the *"ERROR"* response.

3.3 Device modes

RFID device can be in two modes: scanning (SCAN1) and application controlled mode (SCAN0). The default mode (i.e. the device mode after POR) is selected via the device settings. Factory defaults set it to *scanning* mode (SCAN1).

SCAN1 mode is obsolete since v 1.5. See the details below

- SCAN1 - the device continuously scans whether an RFID tag is present or not (actually it queries the tag presence once every N ms, factory default - 1000ms) and reports it to the PC:
  - \r\nSCAN:+<HEX DATA>\r\n when a new tag is detected
  - \r\nSCAN:-<HEX DATA>\r\n when a tag is no longer present
- SCAN0 - the device does not scan for RFID tags on its own, only when a corresponding request arrives.

All commands except LED control, BUZZ control, settings related, and device reboot will fail with ERROR while the device is in the SCAN1 mode.

- (v. *1.3F Dec 17 2018* and above) An extra SCAN2 mode is added. In this mode the device continuously scans for an RFID tag and, once a tag has been identified, prints the data read from it using the specified format: \r\n[formatted data]\r\n. Unlike the SCAN1 mode when the tag is removed, no data is printed. Refer to the FMT command for the format description. All commands except LED control, BUZZ control, settings related, and device reboot will fail with ERROR while the device is in the SCAN2 mode.
- (v. *1.0F Apr 3 2018* and above) DATA is determined by the UID and UID settings. If DATA is the UID, SAK is optionally appended based on the corresponding device setting.
- (prior to v. *1.0F Apr 3 2018*) DATA is the UID + SAK
- Since v1.5 the SCAN1 mode is effectively the same as the SCAN2 (formatted output), the 1 char in the mode is kept for compatibility reasons.

3.4 Error codes

When a tag reports an error, an additional frame describing the error can be returned before the ERROR frame. This additional frame has the following format:

```
1 +CME ERROR: <code>
```

where code is the last error reported by RFID IC. It is a 32bit unsigned integer in decimal format with the following bit fields:

- 0x00000001 - Protocol error
- 0x00000002 - Parity error
- 0x00000004 - Checksum error
- 0x00000008 - Collision
- 0x00000010 - Buffer overflow

- 0x00000020 - Tear event
- 0x00000040 - IC overheated
- 0x00000080 - FIFO write error
- 0x00000100 - Operation timed out
- 0x00000200 - Mifare NAK
- 0x00000400 - Authentication failure
- 0x00000800 - Generic communication error
- 0x00001000 - The TAG returned more data than expected (v1.5+)
- 0x00002000 - The TAG reply integrity error (v1.5+)

Bits 16-31 are reserved for internal purposes and must be ignored.

3.5 CDC/ACM AT Command Set. API Reference.

3.5.1 Device Information

This command queries the product information and firmware version of the device.

Syntax:

Request	ATI\r
Response	<product description> <firmware version/build date>
	S/N <serial number>

Example:

Request	ATI\r
Response	Open-Development RFID Reader (CDC-AT)1.0F Jan 17 2018
	S/N 220333635434B431500280010
	OK

3.5.2 Flush buffers

(*from v1.6*) This command forces the device to flush the internal buffer, holding the data to be sent to the PC.

Syntax:

Request	ATH\r
Response	OK

3.5.3 LED Control

This command controls the state of the light-emitting diode (LED). The -e/-m/-n device revisions do not have any LEDs, therefore this command is not supported. The device will answer with an ERROR to any AT+D.... command

- 0 = off,
- 1 = on
- 2 = blinking

If no state is specified (i.e. nil[1]), the application relinquishes control to the firmware

Syntax:

Request	AT+D1=[0/1/2/nil]\r
Response	OK

Example:

Request	AT+D1=1\r	Turn the LED on
Response	OK	
Request	AT+D1=\r	Return control to the firmware
Response	OK	
Request	AT+D1?\r	Query LED state
Response	+D1=1	LED is on
	OK	

[1] *nil* means *no data transmitted* (ex. AT+D1=).

3.5.4 Buzzer Control

This command controls the state of the buzzer.

- 0 = off
- 1 = on If no state is specified (i.e. `nil`[2]), the application relinquishes control to the firmware.

Syntax:

Request	`AT+B=[0/1/nil]\r`
Response	`OK`

Example:

Request	`AT+B=1\r`	Turn the buzzer on
Response	`OK`	
Request	`AT+B=\r`	Return control to the firmware
Response	`OK`	
Request	`AT+B?\r`	Query the buzzer state
Response	`+B=1`	Buzzer is on
	`OK`	

3.5.5 RFID Mode Control

This command switches the device mode.

Syntax:

Request	`AT+SCAN[0/1/2]\r`
Response	`OK`

Example:

[2] *nil* means *no data transmitted* (ex. `AT+D1=`).

Request	AT+SCANO\r	
Response	OK	Scanning disabled

Request	AT+SCAN?\r	supported since v1.4F[3]
Response	+SCAN=0	Current mode: 0
	OK	

3.5.6 Enable/Disable RF Field

This command switches *on* / *off* the RF field, generated by the IC.

Syntax:

Request	AT+RF=[0/1]\r	1 - switch on
		0 - switch off
Response	OK	

Example:

Request	AT+RF=0\r	
Response	OK	RF field is off
Request	AT+RF?\r	Query the RF field state
Response	+RF=0	RF field is currently off
	OK	

[3] mode 2 supported since v1.3F, mode 1 obsolete since v1.5F.

3.5.7 Inventory scan (all tags)

Requests an inventory scan. Returns a list of UIDs (may be empty) of the tags in the field range. All the discovered tags are scanned and reset after the enumeration completes.

Syntax:

Request	AT+I\r	
Response	+UID=<HEX UID><SAK>	possibly repeated or absent
	OK	

Example:

Request	AT+I\r
Response	+UID=EC6D140708
	+UID=343D7091725D8600
	OK

3.5.8 Inventory Scan (the first tag)

Requests an inventory scan, selects and returns the first detected tag, if any, according to the anti-collision procedure. The tag (if present) will be set as the current one and halted. Note, that if a tag is not present, the device returns an empty success message (i.e OK)

Syntax:

Request	AT+i\r	
Response	+UID=<HEX UID><SAK>	possibly absent
	OK	

Example:

Request	AT+i\r
Response	+UID=EC6D140708
	OK

3.5.9 Inventory Scan (select the next tag)

Halts the currently selected tag and selects the next 13.56MHz tag (if any) according to the anti-collision procedure. The new tag will be set as the current one and halted. Note, that if a tag is not present, the device returns an empty success message (i.e `OK`).

The '-e' revision of the device does not support this command and returns an `ERROR`.

Syntax:

Request	`AT+n\r`	
Response	+UID=<HEX UID><SAK>	possibly absent
	OK	

Example:

Request	`AT+n\r`
Response	+UID=343D7091725D8600
	OK

3.5.10 Select Tag

Select a single 13.56MHz tag by UID for further processing. The tag access commands (read/write block) address the *current* tag. *current* means either the first one according to the anti-collision procedure, or the one selected by this command. Passing an empty UID will deselect the current tag (if any).

ATTENTION: starting from v1.6, **current** should read: 'the tag, selected by the last `AT+i`, `AT+n` or `AT+SELECT=...` command' (**not** the first one selected by the anti-collision procedure)

The '-e' revision of the device does not support this command and returns an `ERROR`.

Syntax:

Request	`AT+SELECT=<HEX UID>\r`	
Response	OK	Tag selected
	ERROR	Tag is not present

Example:

Request	AT+SELECT=343D7091725D86\r	
Response	OK	Tag selected
Request	AT+SELECT=343D7091725D87\r	
Response	ERROR	Tag is not present

3.5.11 Check Tag Presence

Check whether a tag with the specified UID is currently present. Unlike the AT+SELECT=... command this one just checks the tag presence, it does not select the tag and does not make it the current one. The tag being checked will be halted (if present).

The '-e' revision of the device does not support this command and returns an ERROR.

Syntax:

Request	AT+C=<HEX UID>\r	
Response	OK	Tag is present
	ERROR	Tag is not present

Example:

Request	AT+C=343D7091725D86\r	
Response	OK	Tag is present
		The current UID has not changed!

Request	AT+C=343D7091725D87\r	
Response	ERROR	Tag is not present

3.5.12 Get Current Tag Information

Request detailed information about the current tag: UID+SAK, block size, block count, type. UID and SAK are returned in hexadecimal format, BS (block size), BC (block count), T (type) in decimal format.

Recognized tag types

- 0 - Classic 1K
- 1 - Classic 4K
- 2 - Classic Mini
- 3 - Ultralight[4]
- 4 - Ultralight C (UID only)
- 5 - Ultralight C EV1 (640 bits)
- 6 - Ultralight C EV1 (1312 bits)
- 7 - Plus S 2K (SL1)[5]
- 8 - Plus S 4K (SL1)[6]
- 9 - DesFire (UID only)
- 10 - DesFire[7]
- 11 - DesFire[8]
- 12 - Classic 2K
- 13 - Plus X 2K (SL1)[9]
- 14 - Plus X 4K (SL1)[10]
- 15 - Plus X (SL0)[11]
- 16 - Plus X 2K (SL2)[12]
- 17 - Plus X 4K (SL2)[13]
- 18 - Plus X (SL3)[14]

[4] All the Ultralight (i.e. Ultralight, Ultralight EV1) and NTAG tags will be recognized as `Ultralight` (code 3) prior to v1.5

[5] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

[6] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

[7] DESFire is not supported (only the UID can be read) - all the DESFire tags will be identified as 9 since v1.5, as 9, 10, or 11 prior to v1.5.

[8] DESFire is not supported (only the UID can be read) - all the DESFire tags will be identified as 9 since v1.5, as 9, 10, or 11 prior to v1.5.

[9] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

[10] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

[11] *from v 1.5*

[12] *from v 1.5*

[13] *from v 1.5*

[14] *from v 1.5*

- 19 - Plus S (SL0)[15]
- 20 - RFU
- 21 - RFU
- 22 - Plus S (SL3)[16]
- 23 - NTAG213[17]
- 24 - NTAG215[18]
- 25 - NTAG216[19]
- 26 - Ultralight Nano[20]
- 27 - EM41xx compatible 125kHz tag
- 28-254 RFU [21]
- 255 - Unknown

Syntax:

Request	`AT+S\r`	
Response	`+UID=<HEX UID><SAK>,BC=<bc>,BS=<bs>,T=<t>`	
	`OK`	
	`ERROR`	tag did not respond
		or `SCAN1` mode is enabled

Example:

Request	`AT+S\r`
Response	`+UID=EC6D140708,BC=64,BS=16,T=0`
	`OK`

[15] *from v 1.5*

[16] All the Ultralight (i.e. Ultralight, Ultralight EV1) and NTAG tags will be recognized as `Ultralight` (code 3) prior to v1.5

[17] *from v 1.5*

[18] *from v 1.5*

[19] *from v 1.5*

[20] *from v 1.5*

[21] The EM41xx tags do not have a SAK, however the reader returns one for compatibility. A `SAK=0xFF` is returned.

3.5.13 Read Data Block

(*only for Classic, Plus S/X (SL1), Ultralight and NTAGs*)

Read data block from tag memory. The block number must be within the range supported by the tag (ref. the **BC** field in the AT+S\r request), and passed in decimal format. Returns the block number, followed by the actual data in hexadecimal format. Number of bytes in the data equals the tag block size (ref. the **BS** field in the AT+S\r request). A tag must be selected prior to executing this command (either by calling AT+i, AT+n, or AT+SELECT)

The '-e' revision of the device does not support this command and returns an ERROR.

Syntax

Request	AT+R<block number>\r	
Response	+DATA <block number>:<hex data>	
	OK	
	+CME ERROR: <code>	the tag did not respond, reported an error
	ERROR	or SCAN1 mode is enabled

Example

Request	AT+R0\r
Response	+DATA 0:EC6D1407920804009944314230353913
	OK

3.5.14 Write Data Block

(*only for Classic, Plus S/X (SL1), Ultralight and NTAGs*)

Write data block to tag memory. The block number must be within the range supported by the tag (ref. the **BC** field in the AT+S\r request), and passed in decimal format. The data must be passed in hexadecimal format. The length of the data must match the size of a data block as returned in the **BS** parameter (i.e. pass *2*BS* hexadecimal characters).

The '-e' revision of the device does not support this command and returns an ERROR.

Syntax

Request	AT+W<block number>:<HEX DATA>	
Response	OK	
	+CME ERROR: <code>	write or authentication error (if +CME... is reported)
	ERROR	syntax error or SCAN1 mode is enabled (if no +CME...)

Example

Request	AT+W1:000102030405060708090A0B0C0D0E0F\r
Response	OK

3.5.15 Value block

(*supported since v1.6, only for Classic, Plus S/X (SL1)*)

Increment / decrement the value of a "value block" of a Mifare Classic or Plus S/X tag.

The '-e' revision of the device does not support this command and returns an ERROR.

Syntax

Request	AT+V<I/D><block number>:<value>	
Response	OK	
	+CME ERROR: <code>	RW or authentication error (if +CME... is reported)
	ERROR	syntax error or SCAN1 mode is enabled (if no +CME...)

Pass I for increment, D for decrement, block number and value are passed as decimal values. Value must be non-negative.

Example

Request	AT+VI1:42\r
Response	OK

3.5.16 Change Tag UID

Warning: this function is supported starting from *v 1.5*. Not all tags have a re-writable block 0. This is a potentially dangerous operation. The best case scenario - it just won't work. The worst case - you can brick your tag. Proceed at your own risk.

The '-e' revision of the device does not support this command and returns an `ERROR`.

Syntax

Request	`AT+~<HEX UID>`	(*)
Response	`OK`	Success

* UID size **must** match the size of the current UID. UID size changing is **not** supported! All tags except Mifare Classic are **not** supported!

3.5.17 Mifare Plus specific commands

(*supported since v1.6*)

The '-e/-m' revisions of the device do not support these commands and return an `ERROR`.

Authentication

Perform Mifare Plus authentication for the specified block.

Syntax

Request	`AT+N<block>`	
Response	`OK`	Success
	`+CME ERROR: <code>`	Authentication or syntax error
	`ERROR`	

`<block>` must be specified as a decimal number, pass `0` to reset the authentication

Read Block

Read a block of a Mifare Plus tag (assumes a successful authentication procedure for the corresponding block has been performed)

Syntax

Request	AT+M<block number>\r	
Response	+DATA <block number>:<hex data>	
	OK	
	+CME ERROR: <code>	the tag did not respond, reported an error
	ERROR	or SCAN1 mode is enabled

Example

Request	AT+M1\r
Response	+DATA 1:EC6D1407920804009944314230353913
	OK

Write Block

Write data block to tag memory (assumes a successful authentication procedure for the corresponding block has been performed). The block number must be within the range supported by the tag and passed in decimal format. The data must be passed in hexadecimal format. The length of the data must match the size of a data block.

Syntax

Request	AT+E<block number>:<HEX DATA>	
Response	OK	
	+CME ERROR: <code>	write or authentication error (if +CME... is reported)
	ERROR	syntax error or SCAN1 mode is enabled (if no +CME...)

Example

Request	AT+E1:000102030405060708090A0B0C0D0E0F\r
Response	OK

Security Level Switch

Switch the security level of a Mifare Plus tag. The level must be passed as a decimal number.

Syntax

Request	AT+J<level>	
Response	OK	
	+CME ERROR: <code>	write or authentication error (if +CME... is reported)
	ERROR	syntax error or SCAN1 mode is enabled (if no +CME...)

3.5.18 125kHz Tags Commands

Supported by -e/-m revisions only.

Program T55x7

The device can program T55x7 (T5557/67/... and binary-compatible) tags in Em-Marine emulation mode. The tag will be identified by Em-marine compatible readers as a EM4100 tag with the specified UID.

If biphase encoding is used, **please note**, that in the e5555-compatibility mode (default for the T5577 IC) biphase coding is **reversed** compared to the EM4100 standard (mid-bit transition is one, not zero.) The device will therefore try to program the T5577 in extended mode, where a compatible biphase coding is available. This may fail, therefore for compatibility reasons Manchester coding choice is preferable.

Syntax:

Request	AT+@HHHHHHHHHH\r	HH..HH - 5 digits (10 hex characters) of the Customer ID (1 byte) + UID (4 bytes)
Response	OK	

If an inventory scan command has been performed prior to issuing this command and an EM-Marine tag has been detected, the 10 hex characters can be omitted in order to perform a tag clone operation.

Example:

Request	AT+@4201020304\r	
Response	OK	Tag write sequence has been performed

Example 2

Request	AT+i\r	
Response	+UID=1011121314FFOK	EM-Marine tag has been identified
_____	_____	Remove the EM-Marine tag and prepare the tag to be written
Request	AT+@\r	
Response	OK	Tag write sequence has been performed

ATTENTION: The device will not verify by itself that the write operation succeeded (or even that a compatible tag is present). The success or failure of the write command has to be verified by a following inventory scan command.

Program T55x7 with a password

The same as the previous command, but the tag is password-protected against unauthorized access.

Syntax:

Request	AT+xPPPPPPPP,HHHHHHHHHH\r	PP..PP - 32 bit big-endian unsigned integer.
Response	OK	

Comma and the 10 hex digits are optional if a EM-Marine tag has been previously identified (i.e. a clone operation should be performed)

Clear the password protection of a T55x7 tag

ATTENTION. Only the password protection bit will be lifted. The password itself will be left in the tag's memory in plain text, **anyone will be able to read it**.

Please note that the device has no means to verify that the password protection has been disabled, or even that the password was correct.

Syntax:

Request	AT+uPPPPPPPP\r	PP..PP - 32 bit big-endian unsigned integer.
Response	OK	

3.5.19 Device Settings

The following commands change various device settings. They are executed regardless of the device mode. Individual commands change only in-RAM settings. In order to save the current settings to ROM, execute the AT+P command.

Enable/Disable the LED

This command enables/disables the LED. This setting only affects whether the firmware switches the LED when a tag is detected/lost. Regardless of this setting, you can switch the LED on and off using AT+D1... command.

Syntax:

Request	AT+L[0/1/?]\r	? - query the current value
		0 - do not switch LED on/off
		1 - switch LED on/off
Response	OK	

Example:

Request	AT+L?\r	
Response	+L1	LED will be switched on/off
	OK	

Enable/Disable the Buzzer

This command enables/disables the buzzer. This setting only affects whether the firmware will make a *beep* when a tag is detected or not. Regardless of this setting, you can switch the buzzer on and off using AT+B... command.

Syntax:

Request	AT+Z[0/1/?]\r	? - query the current value
		0 - do not make a *beep*
		1 - make a *beep*
Response	OK	

Example:

Request	AT+Z?\r	
Response	+Z1	The device will beep when a tag is detected
	OK	

RFID Receiver Gain

The receiver gain is measured in *dBm*. The valid range is [18;48]dBm. The default value is 33dBm.

The valid gain values are

- 18dBm
- 23dBm
- 33dBm
- 38dBm
- 43dBm
- 48dBm

If any other value is passed, it will be rounded (to the nearest available value greater than the one passed).

Syntax:

Request	AT+G=[18-48]\r	? - query the current value
		+G= - revert to default (33)
Response	OK	

Example:

Request	AT+G=24\r	
Response	OK	The value will be rounded
		The value will not be applied immediately

Request	AT+G?\r	
Response	+G=33	The value (24) has been rounded.

OK

Tag Presence Query Frequency

When in SCAN1 (or SCAN2) mode, the device queries the tag presence every N ms (default value: 1000ms). This command controls the query frequency (i.e. the interval N). Valid range is [100-65535] ms. Values smaller than the minimum one will be accepted and silently increased to fit in the range. Values greater then the maximum one will result in an error.

starting from v1.4: the default value is 200ms.

Syntax:

Request	AT+T[?/<value>]\r	? - query the current value
Response	OK	

Example:

Request	AT+T?\r	
Response	+T1000	The current interval value is 1000ms
	OK	

Request	AT+T1500\r	
Response	OK	The current interval value set to 1500ms

Mifare Authentication Key / Ultralight Password

Mifare Classic tags need an authentication key to be read/written. The following command sets the key (the key type and 6 bytes long key itself), which will be used to authenticate the connected Mifare Classic tags. The default value is FF

Syntax:

Request	AT+K<type><hex data>\r	type can be A or B

Response	OK

Starting from *v 1.5*:

Set the Ultralight EV1 and NTAG's authentication password and query the current tag's PACK

Set the password:

Request	AT+KU<hex data>\r	4 bytes (8 hex digits)
Response	OK	

Query the PACK and perform the authentication:

Request	AT+KP\r	
Response	+P<hex data>	2 bytes (4 hex digits)
	OK	

Starting from *v 1.6*:

Set the Mifare Plus key

Request	AT+KX<hex data>\r	16 bytes (32 hex digits)
Response	OK	

Example:

Request	AT+KA000102030405\r	
Response	OK	Key set to 00..05 type A
Request	AT+K?	
Response	+A000102030405\r	returns current key type and value.
	+UFFFFFFFF	(from v1.5) returns the UL. password
	+XFF..FF	(from v1.6) returns the MF Plus key

	OK	

13.56MHz frequency transceiver control

(for the "-M" revision only) Enable/Disable the high-frequency (13.56MHz) transceiver. Default: enabled.

Note: The 125kHz tags are always scanned **first** unless the low-frequency transceiver is disabled.

Syntax:

Request	AT+h[0/1/?]\r	? - query the current value
		0 - disable
		1 - enable
Response	OK	

Example:

Request	AT+h?\r	
Response	+h1	13.56 transceiver is enabled
	OK	

125kHz frequency transceiver control

(for the "-M" revision only) Enable/Disable the low-frequency (125kHz) transceiver. Default: enabled.

Note: The 125kHz tags are always scanned **first** unless the low-frequency transceiver is disabled.

Syntax:

Request	AT+m[0/1/?]\r	? - query the current value
		0 - disable
		1 - enable
Response	OK	

Example:

Request	AT+m?\r	
Response	+m1	125 transceiver is disabled
	OK	

125kHz Tag Mode Control

(for the "-M/-E" revisions only). This parameter defines how the T55x7 tags will be programmed (i.e. the coding (Manchester/BiPhase) and the speed (32 or 64 transitions per bit))

Syntax:

Request	AT+c [0/1],[0/1]\r	? - query the coding (0 - Manchester, 1 - BiPhase) and the speed (0 - 64tpb, 1 - 32tpb)
Response	OK	

Example:

Request	AT+c?\r	Query the current state
Response	+c0,1	Manchester coding with 32 transitions per bit is selected
	OK	

SAK

(for version *1.0F Apr 3 2018* and above, **not** supported since *1.5F*)

By default Mifare SAK (Select Acknowledge Code) is appended to the UID in the scanning mode. This can be disabled/enabled using the +A command.

Syntax:

Request	AT+A[0/1/?]\r	? - query the current value
		0 - do not append SAK
		1 - append SAK
Response	OK	

Example:

Request	AT+A?\r	
Response	+A=1	SAK will be appended
	OK	

Data selection (scanning mode)

(for version *1.0F Apr 3 2018* and above, **not** supported since *1.5F*)

This setting controls data (UID or block) printed when a new tag is detected. 1 - UID, 0 - block, determined by the Block Number setting

Syntax:

Request	AT+U[0/1/?]\r	? - query the current value
		0 - print block
		1 - print UID (the default)
Response	OK	

Example:

Request	AT+U?\r	
Response	+U=1	UID will be printed
	OK	

Block number selection (scanning mode)

(for version *1.0F Apr 3 2018* and above, **not** supported since *1.5F*)

In scanning mode the device continuously scans whether an RFID tag is present and prints UID or the selected block. This setting controls the block number being printed:

If the block does not exist or cannot be read (ex. due to an authorization failure) nothing is printed.

Syntax:

Request	AT+b[?/<value>]\r	? - query the current value
Response	OK	

Example:

Request	AT+b?\r	
Response	+b=0	Block 0 contents will be printed
	OK	

Request	AT+b1\r	
Response	OK	Block 1 contents will be printed

SCAN2 mode output format

(for version *1.3F Dec 17 2018* and above)

The format string syntax is the same for CDC and HID versions. Refer to the format description in the HID section. The default format is HU*\n. The default mode (i.e. before the first occurrence of a, d, h, or H switch) is the upper-case hexadecimal.

ATTENTION: all occurrences of the "\r" and "\n" sequences (i.e. 2 characters: '\' and 'r' or 'n') will be accepted and printed by the AT+F=... and AT+F? commands respectively, however, all occurrences of \r and \n symbols in the output string during tag scanning will be silently removed due to conflict with the general frame format. The **';'** character is reserved for the multi-command separator and if present in the format input will lead to an error or undefined behavior.

Syntax:

Request	AT+F[?/=<format>]\r	? - query the current format string
Response	OK	

Example:

Request	AT+F?\r	
Response	+F=HU*\n	UID will be printed as upper-case hexadecimal symbols

	OK

Request	AT+F=hB1%*\r	
Response	OK	Block 1 contents will be printed as lower-case hexadecimal symbols

Write settings to ROM

This command saves the current in-RAM setting to ROM, i.e. makes them permanent

Syntax:

Request	AT+P\r
Response	OK

3.5.20 Device Control

The following commands manage the RFID reader device itself:

Device Reboot

Reboot the device

Syntax:

Request	AT+Q\r
Response	OK

Firmware Update

Reboot the device and start the bootloader (this command may not be supported by some devices. If not supported, acts as AT+Q\r).

Syntax:

Request	AT+X\r
Response	OK

4 ODRFID (HID Firmware)

The HID firmware acts as a USB HID Keyboard and a Vendor-Specific HID device. The device scans for new tags and types the UID (or UID+SAK, or the specified block) of the scanned tag by using virtual keystrokes.

The differences in the device revisions are described in the Hardware Revisions section.

4.1 Device modes

The RFID device can be in two modes: the scanning mode and the application controlled mode. The default mode (i.e. then device mode after POR) is selected via the device settings. Factory defaults set it to the *scanning* mode.

- In the scanning mode the device scans for new tags and "types" the UID, UID+SAK and/or block(s) of the scanned tag by using virtual keystrokes. One can control how the data is printed via the device settings (upper case / lower case, carriage return, etc. Starting from the *v1.2F* firmware the device presents full control over the data being printed). The tag presence is queried once every N ms. Factory default is 200ms. Custom HID commands (except the one switching the mode and controlling the device settings) are ignored in the *scanning* mode.
- When the device in the application controlled mode, the user controls the device behavior via custom HID commands.

4.2 HID Reports

This firmware has two reports

- Report 1 is used to send keyboard events when the device is in scanning mode and follows the standard HID Keyboard Report format
- Report 2 is used to configure the device, switching scan modes and query/read/write tags. Both *IN*- and *OUT* reports with ID 2 always consist of 63 bytes (64 including the report ID).

4.3 Custom HID commands

4.3.1 General Description

Custom HID commands are sent and received via HID report with ID 2. The report always consists of 63 bytes. The report ID is always the first byte, therefore the full report length is 64 bytes. The first byte (after the report ID byte) is the command ID. Valid commands are listed below. The next bytes are command parameters. If the command parameters occupy less than 62 bytes, the remaining unused bytes should be treated as a padding - they can have any arbitrary value that has no meaning.

The device answers with a report, where the first byte (after the report ID) is the command id (the same as in the request), the next 4 bytes are the error code (32bit unsigned integer, little endian), and the following bytes are command specific. If the command parameters occupy less than the remaining 58 bytes, the unused bytes should be treated as a padding - they can have any arbitrary value that has no meaning. If the error code is not zero, the following command-specific bytes cannot be trusted and **must** be regarded as meaningless.

The error code has the following bit fields:

- 0x00000001 - Protocol error
- 0x00000002 - Parity error
- 0x00000004 - Checksum error
- 0x00000008 - Collision
- 0x00000010 - Buffer overflow
- 0x00000020 - Tear event
- 0x00000040 - IC overheated
- 0x00000080 - FIFO write error
- 0x00000100 - Operation timed out
- 0x00000200 - Mifare NAK
- 0x00000400 - Authentication failure
- 0x00000800 - Generic communication error
- 0x00001000 - The TAG returned more data than expected (v1.5+)
- 0x00002000 - The TAG reply integrity error (v1.5+)

Bits 16-31 are reserved for internal purposes

The error code of 0 is OK (i.e. *no error*). **ATTENTION:** the data following the error code bytes is valid only if the error code is 0.

Command list:

- 0x01 - Device Information
- 0x02 - LED Control
- 0x03 - Buzzer Control
- 0x10 - Inventory Scan (all tags)
- 0x11 - Inventory Scan (the first tag)
- 0x12 - Inventory Scan (select the next tag)
- 0x13 - Check Tag Presence
- 0x20 - Select Tag
- 0x21 - Tag Details
- 0x22 - Read Data Block
- 0x23 - Write Data Block
- 0x24 - Increment value

- 0x25 - Decrement value
- 0x26 - MF Plus Write Data Block
- 0x27 - MF Plus Security Level Switch
- 0x28 - MF Plus Read Data Block
- 0x29 - MF Plus Authenticate
- 0x2F - Change Tag UID
- 0x30 - Settings: LED
- 0x31 - Settings: Buzzer
- 0x32 - Settings: Gain
- 0x33 - Settings: Query Frequency
- 0x34 - Settings: Mifare Key
- 0x35 - Settings: UID Case (KBD)
- 0x36 - Settings: UID Carriage Return (KBD)
- 0x37 - Settings: SAK (KBD)
- 0x38 - Settings: Scan Type
- 0x39 - Settings: Scan Block Number
- 0x3A - Settings: Output Format
- 0x3F - Settings: Save to ROM
- 0x40 - RFID Antenna State
- 0x41 - RFID Mode Control
- 0x42 - 13.56MHz transceiver control
- 0x43 - 125kHz transceiver control
- 0x60 - T55x7 programming control
- 0x62 - T55x7 programming
- 0x63 - T55x7 password programming
- 0x64 - Clear T55x7 password
- 0x80 - Reboot Device
- 0x81 - Reboot Device (DFU)
- Other values are **RFU**

Abbreviations:

- *IN* - device to host
- *OUT* - host to device

Several commands have one 1byte parameter (*bValue*), that controls a device state or setting. It can be one of the following:

QUERY	0	query the current state/value
ON	1	enable

OFF	2	disable
SWITCH	3	switch state
DEFAULT	4	relinquish control to the firmware
-----	other values	treated as QUERY

4.3.2 Device Information

- Command code: 0x01
- Params (OUT): none
- Params (IN):
  - 4 bytes: error code
  - 1 byte: firmware info length
  - N bytes: firmware info

This command queries the firmware version of the device.

4.3.3 LED Control

- Command code: 0x02
- Params (OUT): *bValue*
  - ON - LED on
  - OFF - LED off
  - SWITCH- blink (250ms)
  - DEFAULT - relinquish control to the firmware
  - other - query state
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current LED state (1 - on, 0 - off)

4.3.4 Buzzer Control

- Command code: 0x03
- Params (OUT): *bValue*
  - ON - buzzer on
  - OFF - buzzer off
  - SWITCH,DEFAULT - relinquish control to the firmware
  - other - query state

- Params (IN):
  - 4 bytes: error code
  - 1 byte: current buzzer state (1 - on, 0 - off)

4.3.5 Inventory Scan (all tags)

Enumerates all the tags currently visible by the reader. All the tags are reset after the scan.

- Command code: 0x10
- Params(OUT): none

One or mode *IN* reports will be sent by the device. Params:

- 4 byte: error code
- 1 byte: *L* - UID length (0 for the last report)
- *L* bytes: UID
- 1 byte: SAK

4.3.6 Inventory Scan (the first tag)

Selects the first tag (according to the anti-collision procedure). The tag (if present) will be set as the current one and halted.

- Command code: 0x11
- Params (OUT): none
- Params (IN):
  - 4 bytes: error code
  - 1 byte: *L* - UID length
  - *L* bytes: UID
  - 1 byte: SAK

4.3.7 Inventory Scan (select the next tag)

Halts the currently selected tag (if any) and selects the next 13.56MHz tag (if any) according to the anti-collision procedure. The new tag will be set as the current one and halted.

The '-e' revision of the device does not support this command and returns an ERROR.

- Command code: 0x12
- Params (OUT): none
- Params (IN):
  - 4 bytes: error code
  - 1 byte: *L* - UID length

  - *L* bytes: UID
  - 1 byte: SAK

4.3.8 Check Tag Presence

Checks whether a 13.56MHz tag with the specified UID is present. If the query succeeds, the tag will **not** be set as the current one. The tag will be halted.

The '-e' revision of the device does not support this command and returns an ERROR.

- Command code: 0x13
- Params (OUT):
  - 1 byte: *L* - UID length (4/7/10)
  - *L* bytes: UID
- Params (IN):
  - 4 bytes: error code
  - 1 byte: *L* - UID length
  - *L* bytes: UID
  - 1 byte: SAK

4.3.9 Select Tag

The 13.56MHz tag with the specified UID will be set as the current one and halted.

The '-e' revision of the device does not support this command and returns an ERROR.

- Command code: 0x20
- Params (OUT):
  - 1 byte: *L* - UID length (0/4/7/10). If *L* == 0, halt the current tag.
  - *L* bytes: UID
- Params (IN):
  - 4 bytes: error code

4.3.10 Get Current Tag Information

ATTENTION: Prior to v1.6, if no tag is selected as the current one, the Inventory Scan (the first tag) command will be executed first. If the current tag fails to respond, the next one will be selected and queried. Starting from v1.6, **current** should read: 'the tag, selected by the last Inventory Scan / Next Tag / Select command' (**not** the first one selected by the anti-collision procedure)

- Command code: 0x21
- Params (OUT): none

- Params (IN):
  - 4 bytes: error code
  - 1 byte: *L* - UID length (0 for the last report)
  - *L* bytes: UID
  - 1 byte: SAK
  - 2 bytes: block count (16bit unsigned integer, little endian)
  - 1 byte: block size
  - 1 byte: tag type

Recognized tag types

- 0 - Classic 1K
- 1 - Classic 4K
- 2 - Classic Mini
- 3 - Ultralight[22]
- 4 - Ultralight C (UID only)
- 5 - Ultralight C EV1 (640 bits)
- 6 - Ultralight C EV1 (1312 bits)
- 7 - Plus S 2K (SL1)[23]
- 8 - Plus S 4K (SL1)[24]
- 9 - DesFire (UID only)
- 10 - DesFire[25]
- 11 - DesFire[26]
- 12 - Classic 2K
- 13 - Plus X 2K (SL1)[27]
- 14 - Plus X 4K (SL1)[28]

[22] All the Ultralight (i.e. Ultralight, Ultralight EV1) and NTAG tags will be recognized as `Ultralight` (code 3) prior to v1.5

[23] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

[24] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

[25] DESFire is not supported (only the UID can be read) - all the DESFire tags will be identified as 9 since v1.5, as 9, 10, or 11 prior to v1.5.

[26] DESFire is not supported (only the UID can be read) - all the DESFire tags will be identified as 9 since v1.5, as 9, 10, or 11 prior to v1.5.

[27] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

[28] Prior to v1.5 - any Plus S 2K/4K or Plus X 2K/4K tag respectively

- 15 - Plus X (SL0)[29]
- 16 - Plus X 2K (SL2)[30]
- 17 - Plus X 4K (SL2)[31]
- 18 - Plus X (SL3)[32]
- 19 - Plus S (SL0)[33]
- 20 - RFU
- 21 - RFU
- 22 - Plus S (SL3)[34]
- 23 - NTAG213[35]
- 24 - NTAG215[36]
- 25 - NTAG216[37]
- 26 - Ultralight Nano[38]
- 27 - EM41xx compatible 125kHz tag
- 28-254 RFU
- 255 - Unknown

4.3.11 Read Data Block

(*only for Classic, Plus S/X (SL1), Ultralight and NTAGs*)

A tag must be selected prior to executing this command (either by calling Inventory Scan (the first tag), Inventory Scan (the first tag), Inventory Scan (the first tag)). The block number must be within the range supported by the tag.

The '-e' revision of the device does not support this command and returns an `ERROR`.

[29] *from v 1.5*
[30] *from v 1.5*
[31] *from v 1.5*
[32] *from v 1.5*
[33] *from v 1.5*
[34] All the Ultralight (i.e. Ultralight, Ultralight EV1) and NTAG tags will be recognized as `Ultralight` (code 3) prior to v1.5
[35] *from v 1.5*
[36] *from v 1.5*
[37] *from v 1.5*
[38] *from v 1.5*

- Command code: 0x22
- Params (OUT): 1 byte block number
- Params (IN):
  - 4 bytes: error code
  - 1 byte: block number
  - 1 byte: N = block size (16 or 4 - equal to the value of the *block size* field returned by the tag info command)
  - *N* bytes: block data

4.3.12 Write Data Block

(*only for Classic, Plus S/X (SL1), Ultralight and NTAGs*)

A tag must be selected prior to executing this command (either by calling Inventory Scan (the first tag), Inventory Scan (the first tag), Inventory Scan (the first tag)). The block number must be within the range supported by the tag. Data size must be equal to the block size value returned by the Inventory Scan (the first tag) command.

The '-e' revision of the device does not support this command and returns an ERROR.

- Command code: 0x23
- Params (OUT):
  - 1 byte: block number
  - 1 byte: data size (*N*)
  - N bytes: data
- Params (IN):
  - 4 bytes: error code
  - 1 byte: block number

4.3.13 Value Block

(*supported since v1.6*, *only for Classic, Plus S/X (SL1), Ultralight and NTAGs*)

Increment / decrement the value of a "value block" of a Mifare Classic or Plus S/X tag.

The '-e' revision of the device does not support this command and returns an ERROR.

- Command code: 0x24 (increment) / 0x25 (decrement)
- Params (OUT):
  - 1 byte: block number
  - 4 bytes: integer value (32bit, little endian)
- Params (IN):

  - 4 bytes: error code
  - 1 byte: block number

4.3.14 Change Tag UID

Warning: this function is supported starting from *v 1.5*. Not all tags have a rewritable block 0. This is a potentially dangerous operation. The best case scenario - it just won't work. The worst case - you can brick your tag. Proceed at your own risk. UID size changing is not supported. All tags except Mifare Classic are **not** supported!

The '-e' revision of the device does not support this command and returns an ERROR.

- Command code: 0x2F
- Params (OUT):
  - 1 byte: UID size (it **must** match the size of the current UID.)
  - N bytes: new UID data
- Params (IN):
  - 4 bytes: error code

4.3.15 Mifare Plus specific commands

(*supported since v1.6*)

The '-e/-m' revisions of the device do not support these commands and return an ERROR.

Authentication

Perform MF Plus authentication for the specified block.

- Command code: 0x29
- Params (OUT):
  - 2 bytes: block number (16 bit, little endian)
- Params (IN):
  - 4 bytes: error code
  - 2 bytes: block number

Read Block

Read a block of a MF Plus tag (assumes a successful authentication procedure for the corresponding block has been performed)

- Command code: 0x28
- Params (OUT):

    - 2 bytes: block number (16 bit, little endian)
- Params (IN):
    - 4 bytes: error code
    - 2 bytes: block number
    - 1 bytes: N = data size (matches the value of the *block size* field returned by the tag info command)
    - *N* bytes: block data

Write Block

Write data block to tag memory (assumes a successful authentication procedure for the corresponding block has been performed). The block number must be within the range supported by the tag. The length of the data must match the size of a data block.

- Command code: 0x26
- Params (OUT):
    - 2 bytes: block number (16 bit, little endian)
    - 1 byte: N = data size (must match the value of the *block size* field returned by the tag info command)
    - *N* bytes: block data
- Params (IN):
    - 4 bytes: error code
    - 2 bytes: block number

Security Level Switch

Switch the security level of a MF Plus tag.

- Command code: 0x27
- Params (OUT):
    - 1 byte: new security level
- Params (IN):
    - 4 bytes: error code
    - 1 byte: new security level

4.3.16 125kHz Tags Commands

Supported by -e/-m revisions only.

Program T55x7

The device can program T55x7 (T5557/67/... and binary-compatible) tags in Em-Marine emulation mode. The tag will be identified by Em-marine compatible readers as a EM4100 tag with the specified UID.

If biphase encoding is used, **please note**, that in the e5555-compatibility mode (default for the T5577 IC) biphase coding is **reversed** compared to the EM4100 standard (mid-bit transition is one, not zero.) The device will therefore try to program the T5577 in extended mode, where a compatible biphase coding is available. This may fail, therefore for compatibility reasons Manchester coding choice is preferable.

- Command code: 0x62
- Params (OUT):
  - 5 bytes: Customer ID (1 byte) + UID (4 bytes)
- Params (IN):
  - 4 bytes: error code

ATTENTION: The device will not verify by itself that the write operation succeeded (or even that a compatible tag is present). The success or failure of the write command has to be verified by a following inventory scan command.

Program T55x7 with a password

The same as the previous command, but the tag is password-protected against unauthorized access.

- Command code: 0x63
- Params (OUT):
  - 4 bytes: password as a 32 bit big-endian unsigned integer.
  - 5 bytes: Customer ID (1 byte) + UID (4 bytes)
- Params (IN):
  - 4 bytes: error code

Clear the password protection of a T55x7 tag

ATTENTION. Only the password protection bit will be lifted. The password itself will be left in the tag's memory in plain text, **anyone will be able to read it**.

Please note that the device has no means to verify that the password protection has been disabled, or even that the password was correct.

- Command code: 0x64
- Params (OUT):
  - 4 bytes: password as a 32 bit big-endian unsigned integer.
- Params (IN):
  - 4 bytes: error code

4.3.17 Device Settings

The following commands change various device settings. The are executed regardless of the scanning mode. Individual commands change only in-RAM settings. In order to save the current settings to ROM, execute the Settings Write command.

Enable/Disable the LED

This command enables/disables the LED. This setting only affects whether the firmware switches the LED when a tag is detected/lost. Regardless of this setting, you can switch the LED on and off using LED Control command.

- Command code: 0x30
- Params (OUT): *bValue*
  - `ON` - enable LED switching
  - `OFF` - disable LED switching
  - other - query the current setting
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current LED control setting (1 - enabled, 0 - disabled)

Enable/Disable the Buzzer

This command enables/disables the buzzer. This setting only affects whether the firmware switches the buzzer on when a tag is detected or not. Regardless of this setting, you can switch the buzzer on and off using BUZZ Control command.

- Command code: 0x31
- Params (OUT): *bValue*
  - `ON` - enable buzzer
  - `OFF` - mute buzzer
  - other - query the current setting
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current buzzer control setting (1 - enabled, 0 - disabled)

RFID Receiver Gain

The recevier gain is measured in *dBm*. The valid range is `[18;48]dBm`. The default value is `33dBm`.

The valid gain values are

- 18dBm

- 23dBm
- 33dBm
- 38dBm
- 43dBm
- 48dBm

If any other value is passed, it will be rounded (to the nearest available value greater than the one passed).

- Command code: 0x32
- Params (OUT): (1 byte) 0 - query the current value, >0 - set new gain value (in dBm)
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current gain value in dBm.

Tag Presence Query Frequency

When in scanning mode, the device queries the tag presence every N ms (default value: 1000ms). This command controls the query frequency (i.e. the interval N). Valid range is [100-65535] ms. Values smaller than the minimum one will be accepted and silently increased to fit in the range. Values greater then the maximum one will result in an error.

starting from v1.4: the default value is 200ms.

- Command code: 0x33
- Params (OUT): (2 bytes - 16bit unsigned little endian integer) 0 - query the current value, >0 - set new interval value (in ms)
- Params (IN):
  - 4 bytes: error code
  - 2 bytea: the current interval value in ms.

Mifare Authentication Key / Ultralight Password

Mifare Classic / Plus tags need an authentication key to be read/written. The following command sets the key (the key type and 6 bytes long key itself), which will be used to authenticate connected Mifare Classic tags, or the 16 bytes long Mifare Plus Key (*from v1.6*). The default value is all FFs

(from *v 1.5*) This command is also used to set the Ultralight EV1 and NTAG's authentication password and query the current tag's PACK.

- Command code: 0x34
- Params (OUT):
  - 1 byte: key type

        * 'A' (ascii code 65) - set the Classic key type to A
        * 'B' (ascii code 66) - set the Classic key type to A
        * 'U' (ascii code 85) - set the Ultralight password (for *v 1.5* and above)
        * 'P' (ascii code 80) - query the current Ultralight card's PACK (for *v 1.5* and above)
        * 'X' (ascii code 88) - set the Mifare Plus Authentication key (for *v1.6* and above)
        * Any other value - query the current key, key type, password (since *v1.5*) and Plus key (since *v1.6*)
    - 6 bytes: key (for Classic) **or**
    - 4 bytes: password (32bit LE - as stored in the corresponding block of the tag) **or**
    - no extra data for PACK query **or**
    - 16 bytes: key (for Plus)
- Params (IN, for 'A', 'B')
    - 4 bytes: error code
    - 1 byte: the current key type - 'A' (ascii code 65) or 'B' (ascii code 66)
    - 6 bytes: the current key
- Params (IN, for 'U')
    - 4 bytes: error code
    - 1 byte: 'U'
    - 4 bytes: the current password (32bit LE)
- Params (IN, for 'P')
    - 4 bytes: error code
    - 1 byte: 'P'
    - 2 bytes: PACK (16bit LE - as stored in the corresponding block of the tag)
- Params (IN, for 'X')
    - 4 bytes: error code
    - 16 bytes: the current key
- Params (IN, for any other value):
    - 4 bytes: error code
    - 1 byte: the current key type - 'A' (ascii code 65) or 'B' (ascii code 66)
    - 6 bytes: the current key
    - 4 bytes: Ultralight password (for *v 1.5* and above)
    - 16 bytes: Plus key (for *v1.6* and above)

UID in Uppercase/Lowercase (keyboard mode)

- Command code: 0x35
- Params (OUT): *bValue*
    - `ON` - print UID in upper case

  - `OFF` - print UID in lower case
  - other - query the current setting
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current case control setting (1 - upper case, 0 - lower case)

125kHz Tag Mode Control

(for the "-M/-E" revisions only). This parameter defines how the T55x7 tags will be programmed (i.e. the coding (Manchester/BiPhase) and the speed (32 or 64 transitions per bit))

- Command code: 0x60
- Params (OUT):
  - 1 byte: *bValue* - `ON`: BiPhase, `OFF`: Manchester, other: query the current state
  - 1 byte: *bValue* - `ON`: 32tpb, `OFF`: 64tpb, other: query the current state
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current coding state (`0` - Manchester, `1` - BiPhase)
  - 1 byte: current speed state (`0` - 64tpb, `1` - 32tpb)

Append/Omit Carriage Return (keyboard mode)

deprecated since version 1.2, not supported since v1.6

- Command code: 0x36
- Params (OUT): *bValue*
  - `ON` - print carriage return after UID
  - `OFF` - do not print carriage return after UID
  - other - query the current setting
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current CR control setting (1 - append, 0 - omit)

Append/Omit SAK (keyboard mode)

deprecated since version 1.2, not supported since v1.5

- Command code: 0x37
- Params (OUT): *bValue*
  - `ON` - append SAK to UID
  - `OFF` - do not append SAK to UID

  - other - query the current setting
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current SAK control setting (1 - append, 0 - omit)

Data selection (scanning mode)

deprecated since version 1.2, not supported since v1.5

This setting controls data (UID or block) printed when a new tag is detected.

- Command code: 0x38
- Params (OUT): 1 byte value
  - `1` - print UID
  - `2` - print the contents of a block determined by the Block Number
  - other - query the current setting
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current value

Block number selection (scanning mode)

deprecated since version 1.2, not supported since v1.5

In scanning mode the device continuously scans whether an RFID tag is present and prints UID or the selected block. This setting controls the block number being printed:

If the block does not exist or cannot be read (ex. due to an authorization failure) nothing is printed.

- Command code: 0x39
- Params (OUT):
  - 1 byte: 0 - query, >0 - set
  - 1 byte: new block number (if `set` mode is selected)
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current value

Set output format

supported since version 1.2

In scanning mode the device continuously scans whether an RFID tag is present and prints the data read from the discovered tag using the specified format.

Format string syntax:

- `a` - switch to ascii mode: output data as ascii symbols. Skips data outside the range 8(backspace), 9(tab), 10(return) and 32(space)-126(~).
- `d` - switch to decimal mode: output data as a decimal (0-255). No zero padding.
- `e` - (only for the -e/-m devices, firmware 2.0 Jan 30 2020 and later) output the EM-Marine UID in decimal format (3 least significant bytes as a 24bit big-endian decimal number padded with zeros to occupy 10 digits). If no EM tag is currently present, the code is ignored. Other device revisions will just output the ascii lettet 'e'.
- `E` - (only for the -e/-m devices, firmware 2.0 Jan 30 2020 and later) output the EM-Marine UID in text format (*xxx.yyyyy*). If no EM tag is currently present, the code is ignored. Other device revisions will just output the ascii lettet 'e'.
- `h` - switch to hexadecimal mode (lower-case)
- `H` - switch to hexadecimal mode (upper-case)
- `U` - output the tag UID. The 'U' switch must be followed by one of the modifiers:
  - `*` - output the whole UID
  - `~` - output the whole UID in reversed order (the last byte first)
  - `n:m` - output bytes n, n+1, ... m (incl.) of the UID. If m<n, output bytes m, m-1, ..., n(incl.). The indices outside the current data range are silently skipped.
  - `n:` - output bytes n, n+1, ... of the UID (till the last byte)
  - `n` - output byte n. (**Attention**: n,m indices start at zero!)
- `BN` - output block N(0-255) from the tag memory. The block numbers range depends on the tag type. If the block does not exist or cannot be read (for example due to the authentication failure), it is silently skipped. The `B` switch must be followed by the block number N, then the '%' symbol and one of the modifiers:
  - `*` - output the whole block (4 or 16 bytes depending on the tag type)
  - `~` - output the whole block in reversed order
  - `n:m` - output the block range (same format as the `U` modifier)
  - `n:` - output part of the block (same format as the `U` modifier)
  - `n` - output the block byte (same format as the `U` modifier)
- `S` - output the SAK of the tag
- `\t` sequence is converted to TAB, `\n` to RETURN.
- any other symbol is printed as is (**no** check whether is is a printable character or not is performed). In order to print reserved symbol ('a', 'd', 'h', etc) escape it with a backslash (\)

The default mode (i.e. before the first occurrence of `a`, `d`, `h`, or `H` switch) is the hexadecimal one. The upper or lower case depends on the Uppercase/Lowercase settings (code 0x35).

Syntax:

- Command code: 0x3A
- Params (OUT):
  - 1 byte: 0 - query, >0 - set
  - max. 32 bytes: format string. If the length is less than 32 bytes, the string **must end with 0** (aka null-terminated string).
- Params (IN):
  - 4 bytes: error code
  - max 32 bytes: the current format string (null-terminated if less than 32 bytes long)

Write settings to ROM

This command saves the current in-RAM setting to ROM, i.e. makes them permanent

- Command code: 0x3F
- Params (OUT): none
- Params (IN): (4 bytes) error code

4.3.18 Device Control

The following commands control the RFID reader device itself:

4.3.19 Enable/Disable RF Field

- Command code: 0x40
- Params (OUT): *bValue*
  - `ON` - enable RF field
  - `OFF` - disable RF field
  - other - query the current mode
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current RF field state (1 - enabled, 0 - disabled)

4.3.20 RFID Mode Control

- Command code: 0x41
- Params (OUT): *bValue*
  - `ON` - enable scan mode
  - `OFF` - disable scan mode
  - `QUERY` - query the current mode
  - other - **error** (will return `INVARG` error)

- Params (IN):
  - 4 bytes: error code
  - 1 byte: current mode (1 - scan mode enabled, 0 - scan mode disabled)

4.3.21 13.56MHz frequency transceiver control

(for the "-M" revision only) Enable/Disable the high-frequency (13.56MHz) transceiver. Default: enabled.

Note: The 125kHz tags are always scanned **first** unless the low-frequency transceiver is disabled.

- Command code: 0x42
- Params (OUT): *bValue*
  - `ON` - enable 13.56 transceiver
  - `OFF` - disable 13.56 transceiver
  - `QUERY` - query the current mode
  - other - **error** (will return `INVARG` error)
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current mode (1 - 13.56 transceiver enabled, 0 - 13.56 transceiver disabled)

4.3.22 125kHz frequency transceiver control

(for the "-M" revision only) Enable/Disable the low-frequency (125kHz) transceiver. Default: enabled.

Note: The 125kHz tags are always scanned **first** unless the low-frequency transceiver is disabled.

- Command code: 0x43
- Params (OUT): *bValue*
  - `ON` - enable 125kHz transceiver
  - `OFF` - disable 125kHz transceiver
  - `QUERY` - query the current mode
  - other - **error** (will return `INVARG` error)
- Params (IN):
  - 4 bytes: error code
  - 1 byte: current mode (1 - 125kHz transceiver enabled, 0 - 125kHz transceiver disabled)

Device Reboot

Reboot the device.

- Command code: 0x80
- Params (OUT): none

- Params (IN): (4 bytes) error code

Firmware Update

Reboot the device and start the bootloader (this command may not be supported by some devices. If not supported, acts as Device Reboot).

- Command code: 0x81
- Params (OUT): none
- Params (IN): (4 bytes) error code

4.3.23 Other commands - RFU

*Params (IN) (4 bytes) error code `0x800` - Generic communication error.

5 ODRFID-485 (AT Firmware)

The device accepts commands and sends a reply via RS485 interface

An AT command set recognized by the device follows the CDC device firmware as close as possible.

5.1 Port settings

Factory defaults are 115200 baudrate, 8N1 (8 bits, 1 stop bit, no parity), device address 0x5F (95). These settings can be changed via the `AT+[` command described below. The new settings (both the current RFID-related settings and the new connection-related settings) are immediately saved in the non-volatile memory and applied after the device reset.

5.2 General Frame Format

The frame can be divided into two levels (close to L2 and L3 in the OSI model)

5.2.1 L2 (UART-RS485)

The *L2* frames format describes how the UART-RS485 data stream is split into frames and how the frames are delivered to the device they are address to. This format is common to both the request and the reply frames.

- The frame starts with a SOF (start of frame) marker 0x7E (dec 126, ascii '~')
- The next byte is the frame payload length (0..254). Note that the payload maximal length is 254, not 255.

- The next byte is the address of the target device (0..254). Note that the 255 address is NOT valid, there must not be a device with such an address.
- The next 0 to 254 bytes are the payload (i.e. the L3 frame)
- The last byte of the frame is the CRC8 (sum with overflow) of the whole frame (excluding the checksum byte itself).

Note that the total length of a frame is 4+`payload length`, the CRC8 is calculated over the whole frame, not just the payload. A frame wihout any payload (`payload length == 0`) is a *valid* frame.

Example:

B0	B1	B2	B3	B4	B5	B6	B7
0x7E	4	0x5F	A	T	I	\r	0xCC

5.2.2 L3 (Commands and data)

The *L3* frames follow the ODRFID (CDC) protocol as close as possible:

- Host-to-Device data must start with an `AT` keyword and can end with a carriage-return character (ASCII code 13 decimal, 0x0d hexadecimal, *"\r"* as C string literal) *Note*: unlike the CDC protocol, the carriage-return character is optional.
- Device-to-Host frames start and end with a carriage-return character followed by a line-feed character (ASCII codes 13,10 decimal, 0x0d,0x0a hexadecimal, *"\r\n"* as C string literal).
- Device-to-Host response to a Host-to-Device request consists of one or more frames, the last one being `\r\nOK\r\n` or `\r\nERROR\r\n`, depending on whether the request/command has been completed successfully or failed.
- Do **not** add any unnecessary white space characters (space, back-space, tab, line-feed, etc.) to the request data, they will **not** be removed by the command-line parser leading to the *"ERROR"* response.

For a complete description of the L3 commands, refer to the CDC documentation. The differences are listed below.

5.2.3 Second LED Control

In addition to the first (green) LED, controlled by the AT+D1... command, this device has the second (red) LED and therefore supports the AT+D2 command.

Syntax:

Request	`AT+D2=[0|1]`	Switch the LED

		0=off, 1=on
Response	OK	

5.2.4 Output Pin Control

(*extra command, compared to the CDC variant*) This command controls the state of the output pin OUT2 (open collector, no pull)

Syntax:

Request	AT+Y=[0|1]	Set the OUT2 state to
		0=off (open circuit), *1=on* (connected to ground)
Response	OK	

Example

Request	AT+Y=1\r	OUT2 state is connected to ground
Response	OK	
Request	AT+Y=0\r	OUT2 state is open circuit
Response	OK	
Request	AT+Y?\r	Query the OUT2 state
Response	+Y=0	OUT2 state is open circuit
	OK	

5.2.5 Input Pin

(*extra command, compared to the CDC variant*) This command reports the state of the input pin (if present)

Syntax:

Request	AT+y?	Request the IN state

Response	+y=0 or +y=1	0 - pulled low, 1 - pulled high
	OK	

5.2.6 UART/RS485 Control

This command queries and sets the UART-related settings (baudrate, bits, parity etc.). The returned values are the ones, saved in the non-volatile memory. They can differ from the actual ones in case they have been altered without a following reboot. The new settings are saved in the non-volatile memory immediately and applied at the next reboot.

Syntax:

Request	AT+[=<a>,<s>,<p>,<b>,<r>,<t>\r	write new settings
Response	OK	

The values passed in the request and returned in the reply:

- <a> - address (0..254), decimal (**ATTENTION**: hex is not accepted, 95 will be interpreted as 0x5F, not as 0x95)
- <s> - stop bits
  - 10 - 1 stop bit
  - 15 - 1.5 stop bits
  - 20 - 2 stop bits
- <p> - parity
  - 0 - none
  - 1 - odd (implemented since v 2.0)
  - 2 - even (implemented since v 2.0)
- <b> - data bits (**only** 8 bits mode is supported, any other value will be accepted, but ignored)
- <r> - baud rate
- <t> - inter-char timeout

Example:

Request	AT+[?\r	
Response	+[=95,10,0,8,115200,10	115200 8N1, address 0x5F, timeout 10ms
	OK	

6 ODRFID-485 (Modbus Firmware)

The device accepts commands and sends a reply via RS485 interface using the Modbus RTU protocol

An AT command set recognized by the device follows the CDC device firmware as close as possible.

The device understands a subset of the Modbus RTU protocol. The supported operations are:

- **Read Input Registers** (function code 4)
- **Read Holding Registers** (function code 3)
- **Preset Single Holding Register** (function code 6)
- **Preset Multiple Holding Registers** (function code 16)
- All the other function codes are not supported and will raise an exception

6.1 Numbering Convention

According to the MODBUS protocol specification, for example, the holding registers occupy the address space 40001 - 49999. However, the transmitted address is the relative one (0 for the 40001, 1 for the 40002, etc.). Here and throughout this document the relative addresses (starting from zero) are used.

6.2 Modbus Registers

6.2.1 Input Registers

- **0** - Returns the size (in bytes) of the internal data pending to be read
- All the other registers are not supported, an attempt to read them will raise an exception

6.2.2 Holding registers

- **0-125** (*0x00 - 0x7D*) - The internal data buffer
  - *write*: send an AT command to the device. The exact starting address is irrelevant (it is recommended to use zero for consistency), as long as the total number of registers holding the AT command fits into this region.
  - *read*: returns the requested portion of the internal data buffer. In order to find out the amount of valid data pending to be read, query the first input register (address 0). The valid data starts at address 0 and occupies *[1st input register value]* bytes (i.e. *ceil([1st input register value]/2)* registers). The device will **not** check whether the requested internal data is valid or not and will always return the requested number of bytes. The internal data read operation **does not clear** the pending data, i.e. the value of the 1st input register will remain the same after the read operation.
- **126** (*0x7E*) - write any value to this register in order to drop the pending data (the 1st input register will be set to 0). This command is **ignored** if an inventory scan is in progress, pending data **not** dropped.

- **127** (*0x7F*) - write any value to this register in order to restart the device
- **128** (*0x80*) - write any value to this register in order to restart the device and put it into the DFU mode. **ATTENTION**: the DFU protocol is not compatible with the Modbus RTU protocol. You will not be able to update the device firmware without disconnecting it from the Modbus line.
- **130** (*0x82*) - **starting from v2.1 only!** (*read-only* in **v2.1**, *read-write* starting from **v2.2**)
  - *read*: the size of the last scanned tag UID
  - *write*: write any value to this register in order to clear (set to zero) registers *130-141*
- **131-141** (*0x83 - 0x8D*) - **starting from v2.1 only!** (*read-only*)
  - 131 - the type of the last scanned tag (ref. Tag Information)
  - 132 - the first byte of the last scanned tag UID
  - 133 - the second byte of the last scanned tag UID
  - . . .

  Note: please consult the p.1 of the Errata
- All the other registers are not supported, an attempt to read/write to them will raise an exception. Registers *130-141* are active **only** in the `SCAN2` mode. They **will not** be updated in the `SCAN0` mode.

6.3 Command sequence example 1:

The default device address (Modbus slave ID) is 95 (*0x5F*)

1. Drop any pending data (you may want to read it first): write *1* to the holding register *126*

   >> 5F06007E0000E4AC

   << 5F06007E0000E4AC

2. Send the AT command (ex. `AT+SCAN0`): write *AT+SCAN0* to the holding registers *0,1,2,3* (i.e. `sizeof(AT+SCAN0)==8` divided by 2. Should the command occupy 7 bytes, it should be padded with 1 zero - *0x00*)

   >> 5F100000000840841542B5343414E304A48

   << 5F1000000004CCB4

   The device returned a success code (*0x10* instead of *0x90* for an exception), therefore there is no need to check for the `\r\nOK\r\n` (or `\r\nERROR\r\n`) response and it is not transmitted by the device.

3. Check that there is no data pending (there should not be any data, this action is shown here for demonstration purposes only)

   >> 5F04000000013CB4

   << 5F0402000010FD

6.4 Command sequence example 2:

1. Drop any pending data (you may want to read it first): write *1* to the holding register *126*

   >> 5F06007E0000E4AC

   << 5F06007E0000E4AC

2. Send the AT command (ex. AT+G?): write AT+G?\0 to the holding registers *0,1,2* (i.e. sizeof(AT+G?)==5 padded with zero (total count 6) divided by 2)

   >> 5F100000000306415 42B473F00A682

   << 5F10000000038D76

   The device returned a success code, now we can query the command result

3. Check the number of bytes pending to be read: read the value of the first input register

   >> 5F04000000013CB4

   << 5F04020009D0FB

   The device reports 9 bytes pending to be read. We need to read 5 holding registers (ceil(9/2)== 5)

4. Read the pending internal data: read the corresponding number of holding registers

   >> 5F030000000588B7

   << 5F030A0D0A2B473D33330D0A30F147

   Note the *garbage* (*0x30*) at the end of data - we have read one extra byte that must be discarded by the processing software. Meaningful data is 9 bytes (0A0D0A2B473D33330D0A, i.e. \r\n+G=33\r\n)

5. **Important**: drop the pending data read (this is the same as the 1st step. Whether to clear the data before or after the command is up to the processing software (as long as the internal buffer does not overflow), but either step 1 or step 4 must be executed)

   >> 5F06007E0000E4AC

   << 5F06007E0000E4AC

6.5 Port settings

Factory defaults are 115200 baudrate, 8N1 (8 bits, 1 stop bit, no parity), device address 0x5F (95). These settings can be changed via the AT+[command described below. The new settings are saved in the non-volatile memory and applied after the device reset.

6.6 General Frame Format

(This applies to the data send and received via the *Preset Holding Registers* and *Read Holding Registers* commands respectively)

- Host-to-Device data must start with an `AT` keyword and can end with a carriage-return character (ASCII code 13 decimal, 0x0d hexadecimal, *"\r"* as C string literal) or zero (0x00) in order to fit data into a number of 16bit registers.

  Note: unlike the CDC protocol, the carriage-return character is optional. Please consult the p.2 of the Errata

- Device-to-Host frames start and end with a carriage-return character followed by a line-feed character (ASCII codes 13,10 decimal, 0x0d,0x0a hexadecimal, *"\r\n"* as C string literal).

- Device-to-Host response to a Host-to-Device request consists of one or more frames.

  Note: unlike the CDC procotol, the `\r\nOK\r\n` and `\r\nERROR\r\n` frames are not transmitted. The success or error code is transmitted in the Modbus reply frame following the Modbus RTU protocol: the MSB of the function code signals the error condition.

- Do **not** add any unnecessary white space characters (space, back-space, tab, line-feed, etc.) to the request data, they will **not** be removed by the command-line parser leading to the *"ERROR"* response.

For a complete description of the L3 commands, refer to the ODRFID (CDC) API reference. Note that there are no `\r\nOK\r\n` and `\r\nERROR\r\n` frames. The differences are listed below.

6.6.1 Inventory scan (all tags)

The command is asynchronous - the scanned tag UIDS will be placed into the buffer (holding registers 0-125) after the scanning process has completed. The device will be unresponsive until while scanning is in progress.

6.6.2 Second LED Control

In addition to the first (green) LED, controlled by the AT+D1... command, this device has the second (red) LED and therefore supports the AT+D2 command.

Syntax:

Request	`AT+D2=[0|1]`	Switch the LED *0=off, 1=on*
Response	Success code as per the Modbus RTU protocol	

6.6.3 Output Pin Control

This command controls the state of the output pin OUT2 (open collector, no pull)

Syntax:

Request	AT+Y=[0|1]	Set the OUT2 state to
		0=off (open circuit), *1=on* (connected to ground)
Response	Success code as per the Modbus RTU protocol	

Example

Request	AT+Y=1\r	OUT2 state is connected to ground
Response	Success code as per the Modbus RTU protocol	
Request	AT+Y=0\r	OUT2 state is open circuit
Response	Success code as per the Modbus RTU protocol	
Request	AT+Y?\r	Query the OUT2 state
Response	+Y=0	OUT2 state is open circuit

6.6.4 Input Pin

(*extra command, compared to the CDC variant*) This command reports the state of the input pin (if present)

Syntax:

Request	AT+y?	Request the IN state
Response	+y=0 or +y=1	0 - pulled low, 1 - pulled high
	OK	

UART/RS485 Control

This command queries and sets the UART-related settings (baudrate, bits, parity etc.). The returned values

are the ones, saved in the non-volatile memory. They can differ from the actual ones in case they have been altered without a following reboot. The new settings are saved in the non-volatile memory immediately and applied at the next reboot.

Attention: when modifying the UART-related settings using this command, all settings will be written (i.e. internally the AT+P command will be called).

Syntax:

Request	AT+[=<a>,<s>,<p>,<b>,<r>,<t>\r	write new settings
Response	Success code as per the Modbus RTU protocol	

The values passed in the request and returned in the reply:

- <a> - address (1..247), decimal (**ATTENTION**: hex is not accepted, 95 will be interpreted as 0x5F, not as 0x95)
- <s> - stop bits
  - 10 - 1 stop bit
  - 15 - 1.5 stop bits
  - 20 - 2 stop bits
- <p> - parity
  - 0 - none
  - 1 - odd (implemented since v 2.0)
  - 2 - even (implemented since v 2.0)
- <b> - data bits (**only** 8 bits mode is supported, any other value will be accepted, but ignored)
- <r> - baud rate
- <t> - inter-char timeout

Example:

Request	AT+[?\r	
Response	+[=95,10,0,8,115200,10	115200 8N1, address 0x5F, timeout 10ms

6.7 Errata

6.7.1 P.1 Holding registers 131-141

Firmware versions affected	*v2.1*, *v2.2*
Action	Reading any number of holding registers starting from *131*
Expected data	The tag type and UID
Data received	Invalid data
Work around	When reading registers 131 and higher, always issue the *Read Holding Registers* command (function code 3) with the starting address value *130*
Example:	Read UID of size 4 from registers 132, 133, 134, 135
WRONG	Read 4 holding registers (function code 3, start address 132, number of registers - 4)
CORRECT	Read 6 holding registers (function code 3, start address 130, number of registers - 6)

6.7.2 P.2 The CR terminator

Firmware versions affected	all versions up to *v2.2*
Action	terminating the `AT...` command with the `\r` character
Expected result	normal frame processing
Actual result	Modbus slave exception
Work around	Do not append the `\r` character to the frame. If the frame needs padding, pad with one zero byte to make the number of characters even.

7 RS485-IO (AT Firmware)

The device accepts commands and sends a reply via RS485 interface

The device understands an AT command set similar to the one of the ODRFID (CDC)

7.1 Port settings

Factory defaults are 115200 baudrate, 8N1 (8 bits, 1 stop bit, no parity), device address 0x5E (94). These settings can be changed via the `AT+[` command described below. The new settings are saved in the non-volatile memory and applied after the device reset.

7.2 General Frame Format

The frame can be divided into two levels (close to L2 and L3 in the OSI model)

7.2.1 L2 (UART-RS485)

The *L2* frames format describes how the UART-RS485 data stream is split into frames and how the frames are delivered to the device they are address to. This format is common to both the request and the reply frames.

- The frame starts with a SOF (start of frame) marker 0x7E (dec 126, ascii '~')
- The next byte is the frame payload length (0..254). Note that the payload maximal length is 254, not 255.
- The next byte is the address of the target device (0..254). Note that the 255 address is NOT valid, there must not be a device with such an address.
- The next 0 to 254 bytes are the payload (i.e. the L3 frame)
- The last byte of the frame is the CRC8 (sum with overflow) of the whole frame (including the checksum byte).

Note that the total length of a frame is 4+`payload length`, the CRC8 is calculated over the whole frame, not just the payload. A frame wihout any payload (`payload length == 0`) is a *valid* frame.

Example:

B0	B1	B2	B3	B4	B5	B6	B7
0x7E	4	0x5F	A	T	I	\r	0xCC

7.2.2 L3 (Commands and data)

The *L3* frames follow the CDC modification protocol as close as possible:

- Host-to-Device data must start with an `AT` keyword and can end with a carriage-return character (ASCII code 13 decimal, 0x0d hexadecimal, *"\r"* as C string literal) *Note*: unlike the CDC protocol, the carriage-return character is optional.
- Device-to-Host frames start and end with a carriage-return character followed by a line-feed character (ASCII codes 13,10 decimal, 0x0d,0x0a hexadecimal, *"\r\n"* as C string literal).
- Device-to-Host response to a Host-to-Device request consists of one or more frames, the last one being `\r\nOK\r\n` or `\r\nERROR\r\n`, depending on whether the request/command has been completed successfully or failed.
- Do **not** add any unnecessary white space characters (space, back-space, tab, line-feed, etc.) to the request data, they will **not** be removed by the command-line parser leading to the *"ERROR"* response.

7.3 AT Command Set. API Reference.

7.3.1 Device Information

ref. ODRFID (CDC)

7.3.2 Flush buffers

ref. ODRFID (CDC)

7.3.3 LED Control

ref. ODRFID (CDC)

7.3.4 Device capabilities

This command provides the information about the number of input/output channels of the device.

Syntax:

Request	AT+C\r
Response	+C=<in>/<out>
	OK

Example:

Request	AT+C\r
Response	+C=3/0
	OK

(this device has 3 inputs and zero outputs)

7.3.5 Input Channel State

Query the current state of an input channel N (N = 1 .. number of channels).

Syntax:

Request	AT+yN?	Query input N state
Response	+yN=[0|1]	
	OK	

Example:

Request	AT+y1?	Query input 1 state
Response	+y1=0	Input 1 is LOW (tied to the ground)
	OK	

7.3.6 Output Channel State

Query or set the state (0 - open circuit, 1 - tied to the ground) of the output channel N (N = 1 .. number of channels).

Syntax:

Request	AT+YN[=0|=1|?]	Set/Query output N state
Response	+YN=[0|1]	
	OK	

Example:

Request	AT+Y2?	Query output 2 state
Response	+Y2=1	Output 2 is 1 (tied to the ground)
	OK	

Request	AT+Y2=0?	Set output 2 state to 0 (open circuit)
Response	OK	Output 2 is now in the open circuit state

7.3.7 Input Channel Pull Configuration

Query or set the pull state (no pull, pull down, pull up) of the input channel.

Attention: not all boards support this setting. In order to save the new setting, issue the `AT+P` (Write Settings to ROM) command.

Pull states:

- 0 - *No pull*
- 1 - *Pull up*
- 2 - *Pull down*

Syntax:

Request	`AT+zN[=0|=1|=2|?]`	Set/Query input N pull state
Response	`+zN=[0|1|2]`	
	`OK`	

Example:

Request	`AT+z1?`	Query input 1 pull state
Response	`+z2=1`	Input 1 is being pulled up
	`OK`	

Request	`AT+z1=0?`	Set input 1 to *no pull*
Response	`ERROR`	The board does not support input pull configuration

7.3.8 Output Channel default state

Query of set the default (startup) state of the output channel N.

Attention: in order to save the new setting, issue the `AT+P` (Write Settings to ROM) command.

States:

- 0 - *open circuit*
- 1 - *tied to the ground*

Syntax:

Request	AT+ZN[=0|=1|?]	Set/Query channel N default state
Response	OK	

Example

Request	AT+Z1=1\r	Set channel 1 default state to 1
Response	OK	
Request	AT+Z2=0\r	Set channel 2 default state to 0
Response	OK	
Request	AT+Z3?\r	Query the channel 3 default state
Response	+Z3=0	Channel 3 default state is open circuit
	OK	

7.3.9 UART/RS485 Control

This command queries and sets the UART-related settings (baudrate, bits, parity etc.). The returned values are the ones, saved in the non-volatile memory. They can differ from the actual ones in case they have been altered without a following reboot. The new settings are saved in the non-volatile memory immediately and applied at the next reboot.

Syntax:

Request	AT+[=<a>,<s>,<p>,<b>,<r>,<t>\r	write new settings
Response	OK	

The values passed in the request and returned in the reply:

- <a> - address (0..254), decimal (**ATTENTION**: hex is not accepted, 95 will be interpreted as 0x5F, not as 0x95)
- <s> - stop bits
  - 10 - 1 stop bit
  - 15 - 1.5 stop bits

  - 20 - 2 stop bits
- <p> - parity
  - 0 - none
  - 1 - odd
  - 2 - even
- <b> - data bits (**only** 8 bits mode is supported, any other value will be accepted, but ignored)
- <r> - baud rate
- <t> - inter-char timeout

Example:

Request	AT+[?\r	
Response	+[=95,10,0,8,115200,10	115200 8N1, address 0x5F, timeout 10ms
	OK	

7.3.10 Write settings to ROM

ref. ODRFID (CDC)

7.3.11 Device Control

ref. ODRFID (CDC)

8 Hardware Revisions

Device	Revision	Firmware Code	Features	Remarks
ODRF	nil	F	13.56MHz	Supports Mifare tags, LED/Buzzer indication
	-M	m	13.56MHz / 125kHz	Supports Mifare tags (Mifare Plus is supported only in Classic-compatibility mode), Em-Marine (read), T5577 (write), Buzzer indication
	-N	n	13.56MHz	Supports Mifare tags (full Mifare Plus support), Buzzer indication
	-E	e	125kHz	Em-Marine (read), T5577 (write), Buzzer indication

Important information. Please read carefully: * All the revisions have the 'CDC' variant, and the 'HID' variant. Generally you can change the CDC/HID variant by upgrading the firmware (even if the firmware version is the same). However, the *-M*, *-E*, and *-N* revisions **cannnot** be simply "reflashed" from one to another. Please verify that you are flashing a compatible firmware during the upgrade process. Flashing *-M* device with *-E* firmware and so on will **brick** your device. In this case you will have to send the device to an authorized service center for restoration. * The *-M* and the *-E* revisions are able to read the EM41xx compatible 125kHz tags with Manchester and BiPhase coding and 32 or 64 transitions per bit. EM4100 specification states that the "*middle-bit transition is zero*" variant of the BiPhase coding is used. The device will report read success on both variants (i.e. "*middle-bit transition is zero*" and "*middle-bit transition is one*").

9 Documentation Revisions

Revision	Date	Remarks
1.0	2018-01-22	Initial Version
1.1	2018-04-27	Protocol update
		Support UID / Block selection in scan mode
1.2	2019-04-07	Add RS485 modification
1.3	2019-08-20	Add new features supported by the v1.6 firmware
1.4	2019-10-29	Add new auxiliary device: RS485-IO
1.5	2020-01-27	Support for the 125kHz-enabled devices
1.6	2020-12-08	MODBUS firmware v2.1 description
1.7	2021-03-16	fixed HID 125kHz related commands

10 About

Open Developent LLC open-dev.ru

MkDocs theme: *mkdocs-material* by Martin Donath

Pandoc theme: *EisVogel* by Pascal Wagler and John MacFarlane