Advanced RFID Card Reader Writer

RFIDAX Device Series

USB NFC Reader Writer

Reference Manual V1.00

info@rfidax.com
www.rfidax.com

Revision History

Date	Changes	Version
2025-02-05	• Initial Release	0.01

info@rfidax.com
www.rfidax.com

List of Figures

List of Tables

1.0 Introduction

RS485-RW Series TC-RW Series TC-RD Series MUSB-RD Series MUSB-RW Series HID-TC Series HID- MUSB Series

RFID readers and writers developed by RFIDAX are based on 13.56 MHz contactless technology and feature superior capabilities with integrated NFC functionality.
The device developed by RFIDAX fully supports the ISO/IEC 14443 Type A protocol and the protocols of the MIFARE family, ensuring seamless compatibility with NFC technology.

RFIDAX devices operate in the 13.56 MHz band and represent the fifth generation of such technology. These devices perform high-tech RFID reading and writing operations. All devices designed by RFIDAX within the 13.56 MHz frequency band are contactless readers and writers.

Equipped with HID support, the system provides fast and driverless communication via USB, enabling easy plug-and-play integration. Additionally, thanks to the standard USB protocol, high-speed data transfer is supported, ensuring a flexible infrastructure suitable for various communication requirements. The independent operation of HID and standard USB protocols offers an optimized solution for different application scenarios.
The system also enables reliable, long-distance data transmission through the RS485 physical layer. With its differential signal structure, RS485 ensures stable and reliable communication even in noisy environments, while the USB interface provides fast data transfer and firmware update support.
Compatible with a wide range of applications, including industrial automation, building management systems, energy monitoring and control, smart city infrastructures, access control systems, and financial data processing, the system offers customizable data packets and a flexible command architecture through the Modbus protocol and a CRC-based custom communication structure. This versatile communication infrastructure delivers a powerful, reliable, and flexible solution for commercial and industrial applications.

The device supports multiple communication interfaces, including USB, RS485, Type-C, and mobile communication, providing a versatile and flexible architecture. With configurable addressing, adjustable communication speed, and a flexible communication protocol, it ensures high compatibility and ease of use. Its fast restore-to-previous-settings mode guarantees a reliable and stable operating environment under all conditions. It also allows selection between active and passive CRC devices for data integrity and supports different operational modes such as Basic, Advanced, Enterprise, and Custom to meet diverse application needs.

Additionally, the device offers intelligent support for two NFC modes: reader/writer and keyboard emulation. Thanks to its flexible design, it can be seamlessly integrated into a wide range of applications, including e-payment systems, access control, and transportation ticketing. Its advanced capabilities make it a reliable, powerful, and adaptable solution for commercial and industrial use cases.
RFIDAX devices stand out in the industry with superior communication speed, the ability to support multiple communication protocols, compatibility with various physical topologies, and an advanced technology infrastructure in the 13.56 MHz card reader and writer market.

With its cutting-edge technology, it distinguishes itself from competitors and holds a leading position in the market.

2.0 Features

☑Performance

- RS485 Communication Speed: Up to 115.2 kbps (Can be increased if required)
- USB Communication Speed: Up to 2000 kbps (Can be increased if required)
- ISO 14443A Read/Write Speed: Up to 424 kbps (Can be increased if required)
- Real-time modification of device address, communication speed, and communication protocol
- Rapid and reliable restoration to factory settings during operation (ideal for defense and space industry applications)
- Programmable LED and buzzer structure

☑Support various contactless card type

- ISO 14443 Type A (Parts 1-4)
- MIFARE® (T=CL)
- MIFARE Classic
- MIFARE Ultralight
- MIFARE DESFire EV1
- MIFARE Plus
- MIFARE Mini
- NFC Tags

☑Support various NFC features

- Card Reader/Writer Mode
- Card Reader Only Mode
- Keyboard Mode

☑Security

- Advanced Software Cryptography: Software-based cryptography support for high-level security.
- Free USB Firmware Updates: Easy updates for the latest features and improvements.

☑Reliability and Durability

- Compliance with International Standards: The device is designed to meet international regulations and standards, making it suitable for military systems.
- Mean Time Between Failures (MTBF): Provides a reliable operational lifespan of up to 750,000 hours.
- Short Circuit and ESD Protection: Integrated protection ensures electrical safety.
- EMI-Compliant Design: Protects against electromagnetic interference.
- Use of ASA, ABS, and Carbon Materials: Manufactured with high-quality materials to enhance durability and ensure long-term usage.
-

☑Compatibility

- Support for all major operating systems
  - Windows®, Linux®, macOS, Android™

3.0 RFIDAX Series Architecture

3.1 Reader Writer Block Diagram

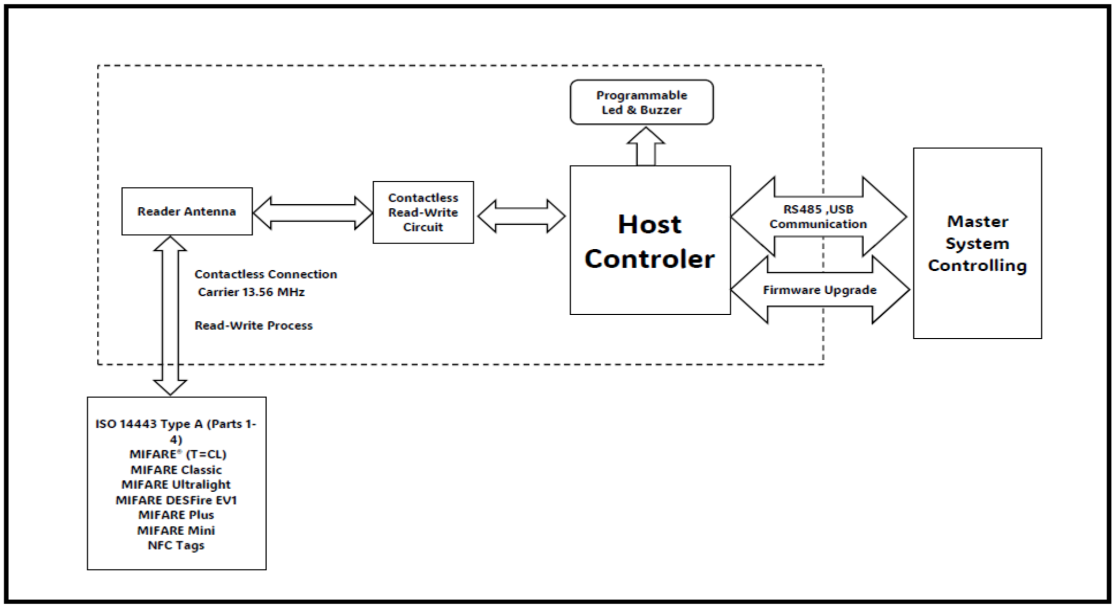

Figure 1 RFIDAX Reader Block Diagram

3.2 Block Diagram of Communication Protocol Structure

RFIDAX devices support various physical communication layers such as RS485, Mini USB, and Type-C, offering a compre hensive connection infrastructure. The device is equipped with advanced communication protocols specifically designed for use in industrial environments and the defense industry.

These protocols can be adjusted to desired speeds based on user requirements, and speed and protocol changes can be made during runtime. This unique feature provides a significant advantage over competing products, making the system both flexible and a high-performance solution.

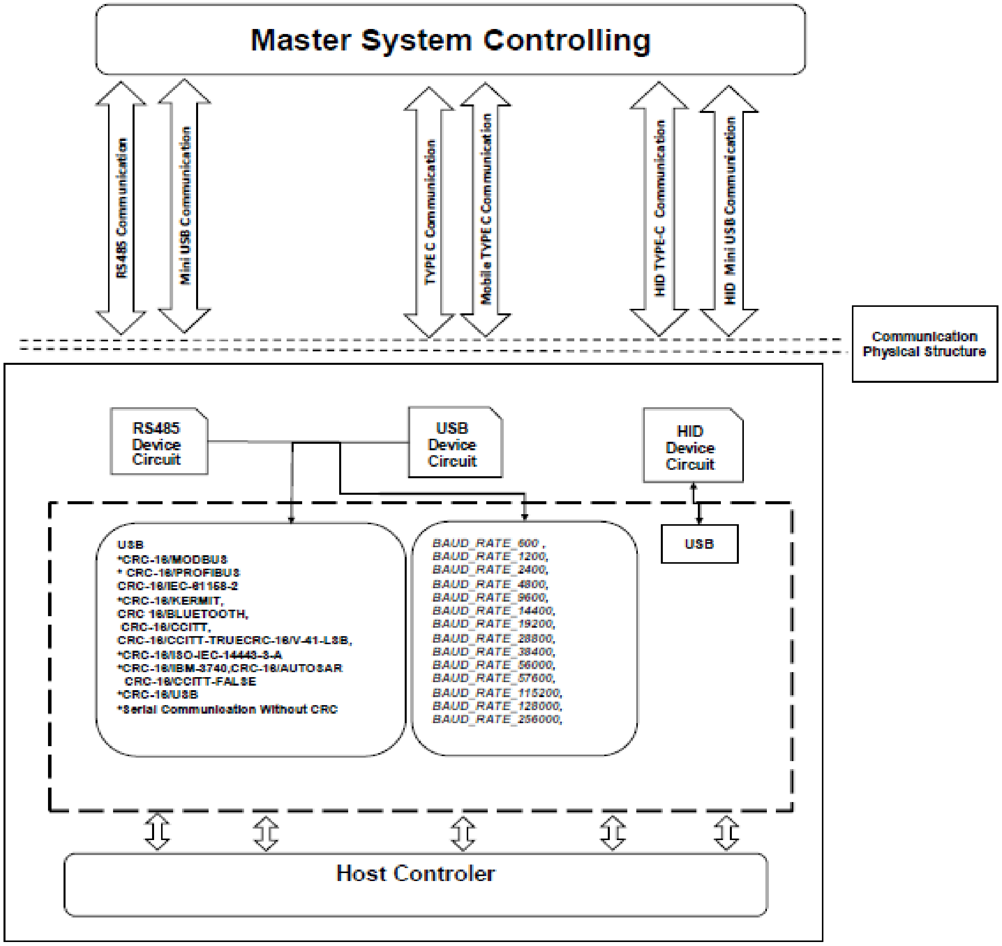

Figure 2 RFIDAX Devices Communication Block Diagram

4.0 RFIDAX Device Models and Device Numbers

The table below provides the model names and abbreviations for RFIDAX devices. Additionally, descriptive information is included. You can select a device from this table based on the desired features and modes.

Model Series	Model Number	Device Description
Mini USB Series	MUSB-RD	A device that performs reading operations using a Mini USB connecto
	MUSB-RW	A device that performs both reading and writing operations using a Mini USB connector. Supports various software protocols.
	HID- MUSB	A HID device that uses a Mini USB connector. Functions like a keyboard.
Type-C Series	TC-RW	A device that performs both reading and writing operations using a Type-C connector. Supports various software protocols.
	TC-RD	A device that performs reading operations using a Type-C connector.
	TC-Mobile	A device that performs both reading and writing operations using a Type-C connector. Supports various software protocols.
	HID- TYPEC	A HID device that uses a Type-C connector. Functions like a keyboard.
RS485 Series	RS485-RD	A device that performs reading operations using an RS485 connector.
	RS485-RW	A device that performs both reading and writing operations using an RS485 connector. Supports various software protocols.

Table 1 Device Model Numbers and Model Information

5.0 Hardware Design

5.1USB

The HID USB and regular USB models of RFIDAX devices are connected to the computer via USB in accordance with the USB standard.

5.1.1. Communication Parameters

RFIDAX connects to a computer via USB in compliance with the USB 2.0 Standard. RFIDAX devices operate in full-speed mode, with a speed of 12 Mbps.

Pin	Signal	Function
1	VBUS	Provides a +5 V power supply for the device.
2	D-	Differential signal line used for data transmission between the RFIDAX device and the computer.
3	D+	Differential signal line used for data transmission between the RFIDAX device and the computer.
4	GND	Reference voltage level for the power supply.

Table 2 USB Interface Wiring

Note: *To ensure proper operation of the device via the USB interface**, the device driver must be installed.***

5.2 RS485

RFIDAX's RS485-RW and RS485-RD series devices can communicate with a computer or device control units through the RS485 physical layer. RS485 devices require a USB-to-RS485 converter for communication.
The communication method is **half-duplex**.

5.2.1. Communication Parameters

The communication speed over the RS485 physical layer may vary and is adjustable. The configurable communication speeds are **600bps, 1200bps, 2400bps, 4800bps, 9600bps, 14.4kbps, 19.2kbps, 28.8kbps, 38.4kbps, 56kbps, 57.6kbps, 115.2kbps, 128kbps, and 256kbps**.
This is a distinguishing feature compared to competing products.

Pin	Signal	Function
1	VBUS	Provides a +24V power supply for the device. (Models with an adjustable +12V option are available upon request.)
2	RS485 -A	Differential signal line used for data transmission between the RFIDAX device and the computer.
3	RS485 -B	Differential signal line used for data transmission between the RFIDAX device and the computer.
4	GND	Provides the reference voltage level for the power supply.

Table 3 RS485 Interface Wiring

Note: When facing the device, the topmost pin **(Pin 1)** is the **VCC pin.** The bottommost pin, which is closest to the LED, is the **GND pin.**

5.3 Type-C Connection

RFIDAX-branded devices, including **HID TYPE-C, standard TYPE-C, and mobile models**, connect to a computer via USB in compliance with the Type-C standard (**TC-RW, TC-RD, TC-Mobile**).
The physical layer of the connection is **Type-C**.

5.3.1. Communication Parameters

RFIDAX connects to computers and mobile devices via USB in compliance with **USB 3.0 and USB 2.0 Standards**. It operates in **full-speed mode**, with a data transfer rate of **12 Mbps**.

Pin	Sinyal	Görev
A4	VBUS	The power line of the controlling device may vary. RFIDAX is designed for this and can self-optimize (**min. +5V, max. 24V**). It adapts to the controlling device. Complies with standard PC power requirements.
B9	VBUS	The power line of the controlling device may vary. RFIDAX is designed for this and can self-optimize (**min. +5V, max. 24V**). It adapts to the controlling device. Complies with standard PC power requirements.
A9	VBUS	The power line of the controlling device may vary. RFIDAX is designed for this and can self-optimize (**min. +5V, max. 24V**). It adapts to the controlling device. Complies with standard PC power requirements.
B4	VBUS	The power line of the controlling device may vary. RFIDAX is designed for this and can self-optimize (**min. +5V, max. 24V**). It adapts to the controlling device. Complies with standard PC power requirements.
A6	DP1	Differential signal line used for data transmission between the RFIDAX device and the computer.
B6	DP2	Differential signal line used for data transmission between the RFIDAX device and the computer.
A7	DN1	Differential signal line used for data transmission between the RFIDAX device and the computer.
B7	DN2	Differential signal line used for data transmission between the RFIDAX device and the computer.
A5	CC1	Used only in RFIDAX mobile devices.
B5	CC2	Used only in RFIDAX mobile devices.
A1	GND	Provides reference voltage level for the power supply.
B1	GND	Provides reference voltage level for the power supply.
A12	GND	Provides reference voltage level for the power supply.
B12	GND	Provides reference voltage level for the power supply.
SHIELD	SHIELD	Acts as a shield for the device.

Table 4 TYPE-C Interface Wiring

Note: *To ensure proper operation of the device via the* ***TYPE-C interface,*** *the* ***device driver must be installed.***

5.4 Contactless Smart Card Interface

The interface between RFIDAX devices and contactless cards operates in compliance with **ISO 14443 specifications**. However, to enhance the practical functionality of RFIDAX, **data manipulation, certain restrictions, or improvements may be applied**.

5.4.1. Carrier Frequency

The carrier frequency of RFIDAX is **13.56 MHz**.

5.4.2. Card Polling

RFIDAX **automatically polls contactless** cards within its field. It **supports ISO 14443 Type A (Parts 1-4), MIFARE® (T=CL), MIFARE Classic, MIFARE Ultralight, MIFARE DESFire EV1, MIFARE Plus, MIFARE Mini, and NFC Tags.**

info@rfidax.com
www.rfidax.com

5.5 User Interface

5.5.1. Buzzer ve LED

The **monotone programmable buzzer** and **programmable LEDs** are used to indicate the status of the contactless interfaces. The **blue and green LEDs**, along with the buzzer, indicate the device and communication status.
The **green LED** is available only in **hardware revision 1**, while the **blue LED** is used in devices **other than revision 1**.
The **red LED** is present in all versions.

Device Status	Buzzer	Red and Blue LED
1. Reader plugged in	Off	🔴
2. Standby mode (Contactless polling active, no card detected)	Off	🔴
3. Standby mode (Polling disabled)	Off	🔴
4. Contactless card tapped	1 Beep	🔵
5. Contactless card remains in the reading area	Off	🔴
6. Contactless card removed	Off	🔴
7. Contactless card communicating	Off	Fast Blinking >> 🔵 >> 🔴 >> 🔵>>🔴

Table 5 Buzzer and LED Indicators

6.0 Device Software Commands and Configuration

This section will describe the **software command structure** of the device, its operational algorithm (as needed), and the structure of incoming and outgoing data.

The following topics will be explained for proper device operation:

- The data structure to be sent,
- Responses from the device and their meanings,
- System monitoring structure and error handling,
- Performing all operations related to RFID cards supported by the device,
- Device modes.

At first glance, the **device commands and configuration** may seem complex, but they follow a specific algorithm. The system is **highly simple and development-friendly**.

While explaining the command structure of the device, **free terminal software** will be used.

The purpose of using free terminals is to help you easily **integrate RFIDAX devices into your software**.

Hexadecimal values will be used throughout the entire device communication process.

To better understand the **command structure**, initial definitions and general system information will be provided.

6.1 Device Modes and Descriptions

The device modes and their descriptions are provided below.

The function types and tasks of this command structure may vary depending on **operation modes**.

These variations occur in the form of **activation or deactivation** of functions.

The available **operation modes** are listed below

Device Mode	Description
MODE_BASIC	Optimized for corporate use. This mode only performs **RFID card reading and basic operations.** Suitable for commercial applications.
MODE_ADVANCED	Performs **both reading and writing operations**. All features are enabled. Used in **commercial, defense, and aerospace industries**.
MODE_CUSTOM	A **customer-specific mode**. Designed for special corporate clients.
MODE_ENTERPRISE	Optimized for corporate use and offers **advanced features for commercial applications.**

Table 6 Device Modes and Descriptions

CUSTOM Mode is a completely **custom-designed mode tailored to the specific needs of customers**. This mode is available only for clients who contact **RFIDAX**. Features are adjusted based on customer requirements, making it a **fully personalized mode**.

ENTERPRISE Mode, on the other hand, **utilizes existing features without modifying the structure** and is designed for corporate clients. It is essentially a **restricted version of ADVANCED Mode**, where the core functionalities remain unchanged.

This document **will not cover CUSTOM Mode**.

� Throughout the documentation, it is specified which features are **active in each mode**.

6.2 Data Integrity Modes and Descriptions

This mode determines whether **CRC (Cyclic Redundancy Check)** is active or not. The available modes are listed in the table below.

Veri Bütünlüğü Modları	Veri Bütünlüğü Modları Açıklamaları
DATA_INTEGRITY_CHECK	**Optimized for corporate use**. Indicates that **CRC is active. (**Used in **MODE_ADVANCED and MODE_ENTERPRISE).**
NO_DATA_INTEGRITY_CHECK	Optimized for corporate use. Indicates that **CRC is not active**. (Used in MODE_BASIC).

Table 7 Data Integrity Modes and Descriptions

6.2.1 When Data Integrity is Inactive

In this mode, **CRC is not active**. **MODE_BASIC** operates with this mode enabled.

6.2.2 When Data Integrity is Active

When **DATA INTEGRITY CHECK is** enabled**, various CRC modes** become active.
MODE_ADVANCED and MODE_ENTERPRISE use this mode actively.
The **CRC modes listed below** provide advanced and professional solutions.
Users can select **any CRC mode** based on their needs.
By default, when CRC is active, the system uses the PROTOCOL_CRC16_CCITT structure.
All CRC modes use CRC-16.

CRC Mode	Supported Subprotocols
PROTOCOL_CRC16_USB	CRC-16/USB
PROTOCOL_CRC16_PROFIBUS	CRC-16/PROFIBUS CRC-16/IEC-61158-2
PROTOCOL_CRC16_MODBUS	CRC-16/MODBUS MODBUS
PROTOCOL_CRC16_KERMIT,	CRC-16/KERMIT CRC-16/BLUETOOTH CRC-16/CCITT CRC-16/CCITT-TRUE CRC-16/V-41-LSB CRC-CCITT KERMIT
PROTOCOL_CRC16_CCITT	CRC-16/IBM-3740 CRC-16/AUTOSAR CRC-16/CCITT-FALSE
PROTOCOL_CRC16_ISO_IEC14443_A,	CRC-16/ISO-IEC-14443-3-A CRC-A

Table 8 CRC modes and sub-CRC modes supported by the modes

6.3 Communication Types of Devices

The communication types of the devices are listed below. These types are categorized based on their **hardware structures**. Accordingly, **RFIDAX devices are named based on this structure**, and model names and numbers are assigned accordingly.

All device modes support these communication types.

Communication Type	Description
COMM_USB	A device that uses **USB hardware**. Available for sale and active.
COMM_RS485	A device that uses **RS485 hardware**. Available for sale and active.
COMM_TYPE_C	A device that uses **Type-C hardware**. Available for sale and active.
COMM_MOBILE	A device that uses **Type-C hardware**. Available for sale and active.
COMM_ETHERNET	Under development, **not available for sale**.
COMM_WIFI	Under development, **not available for sale.**

Table 9 Communication Types and Their Descriptions

6.4 Device Status Messages, Error Codes, and Descriptions

This section lists the possible error codes, their explanations, and potential solutions for issues that may occur in the RFIDAX device.

Important Information:

- All error codes are defined with **4-byte spacing**, making it easier to add new codes in the future.
- Error codes are displayed in **HEX format**, and the device will return them in this format during communication.
- Each error code includes a **"Possible Solution"** section, which provides the necessary steps for users to resolve the issue.

6.4.1 General System Status and Errors

These error codes represent fundamental errors that may occur during the device's general operation and communication.

Error Code	Error Description	Possible Solution
0x0000	SUCCESS – Operation successful.	No error, the operation was completed successfully.
0x0001	ERR_UNKNOWN_COMMAND – Unknown command.	An invalid command may have been sent; check the supported commands.
0x0002	ERR_CRC – CRC error.	The data packet may have been received incorrectly, it may not be in the correct CRC mode, or the master device may have miscalculated the CRC.
0x0003	ERR_ADDRESS_MISMATCH – Device address mismatch.	Check that the command is being sent to the correct device.
0x0004	ERR_BUFFER_OVERFLOW – Memory overflow	Check the length of the command data and resend it appropriately.

Table 10 General System Errors and Description

6.4.1.1 Additional Explanation for General System Errors

ERR_ADDRESS_MISMATCH: This error occurs when the device address is incorrect. Check the device address.
ERR_BUFFER_OVERFLOW: When this error occurs, check the buffer size of the data sent to the RFIDAX device. The buffer size **must not exceed 1024 bytes**. If the buffer size is greater than **1024 bytes**, the ERR_BUFFER_OVERFLOW error will be returned.

6.4.2 Tieout and Data Errors

Error Code	Error Description	Possible Solution
0x0008	ERR_TIMEOUT – Timeout.	No response received from the device; check the connection.
0x000C	ERR_INVALID_DATA – Invalid data	Verify the accuracy of the transmitted data.

Table 11 Timeout and Data Errors and Descriptions

6.4.2.1 Additional Explanation for Timeout and Data Errors

ERR_TIMEOUT: This error occurs rarely. Please ensure that the command sent to the device is within the correct execution time.
ERR_INVALID_DATA This error occurs rarely. Please check the structure of the command being sent to the device.

6.4.3 LED and Indicator-Related Errors

This set of errors indicates **issues related to LED status and control.**

Error Code	Error Description	Possible Solution
0x0010	ERR_INVALID_LED_STATE Invalid LED state.	Ensure the LED is correctly checked for ON or OFF status.
0x0014	ERR_LED_NUMBER_INVALID – Invalid LED number.	Verify the LED ID being used is correct.
0x0018	ERR_INVALID_TOGGLE – Invalid toggle operation.	Check if the LED has a valid state.

Table 12 LED and Indicator-Related Errors and Descriptions

6.4.3.1 Additional Explanation for LED and Indicator-Related Errors

ERR_INVALID_LED_STATE This error is related to an incorrect LED state. In the **RFIDAX** device, LEDs are controlled in two states. The LED state is defined in **hex** as **0x00 (OFF) or 0x01 (ON)**. Any other value will return this error.
ERR_LED_NUMBER_INVALID his error occurs when an invalid LED number is used. In the RFIDAX device, LED numbers are defined in hex as 0x00 to 0x03. Any other value will return this error.

Note: The LED number 0x02 is reserved for Custom Mode and is not used in BASIC or ADVANCED modes.

6.4.4 RFID Card Read/Write Errors

This error set includes issues that may occur during **RFID card reading and writing operations**. To ensure the card is read correctly and data is processed properly, these error codes should be carefully reviewed.
If the card is not detected, the UID cannot be read, or authentication fails, you can refer to the error codes in this section to identify the cause of the issue. These errors are generally related to the **physical position of the card, authentication keys, or block access permissions**.
Some errors may be caused by software settings, so ensure that the system is correctly configured. Errors related to writing data to Flash memory can affect the system's ability to store information, so necessary precautions should be taken.
If you encounter errors during RFID card reading or writing, follow the error codes and possible solutions below to troubleshoot the issue.

Error Code	Error Description	Possible Solution
0x001C	ERR_CARD_UID_READ_FAILED Card UID could not be read.	Try scanning the card again or check the connection.
0x0020	ERR_CARD_NOT_FOUND – Card not found.	Ensure that the card is properly positioned within the reader's field.
0x0024	ERR_ATQA_VALUE_NOT_READ – ATQA value could not be read.	Verify the card type and ensure it is supported..
0x0028	ERR_CARD_NOT_SELECTED — RFID card could not be selected.	Try positioning the card differently or scan it again.
0x002C	ERR_INVALID_KEY_TYPE – Invalid key type.	Ensure that the correct key is being used
0x002D	ERR_AUTHENTICATION_FAILED Block authentication failed.	Check whether the authentication key and process are correct
0x0030	ERR_UNKNOWN_RFID_SUBCOMMAND – Unknown RFID subcommand.	Verify the supported subcommands.
0x0034	ERR_INVALID_BLOCK_RANGE – Invalid block range.	Ensure that the block number is correct
0x0038	ERR_INVALID_AUTH – Authentication failed.	Verify user authorization and key validity
0x004C	ERR_BLOCK_READ_FAILED – Block read error.	Ensure the card is properly in contact with the reader.
0x0050	ERR_DATA_LENGTH_EXCEEDED – Data length exceeded.	Reduce the amount of data being sent.
0x0054	ERR_BLOCK_WRITE_FAILED – Block write error.	Verify whether the card is writable
0x0058	ERR_FLASH_KEY_OPERATION – Invalid Flash key operation.	Ensure that the Flash write operation is valid.
0x0060	ERR_FLASH_ID_RECORD_FAILED – ID record operation failed.	Check if there is enough space in Flash memory.

Table 13 RFID Card Read/Write Errors and Descriptions

6.4.4.1 Additional Explanation for RFID Card Read/Write Errors

ERR_CARD_UID_READ_FAILED: Indicates that the card is detected, but the **UID could not be read**. Check the card's condition and ensure it is correctly positioned within the reading area.

ERR_UNKNOWN_RFID_SUBCOMMAND: The device's command structure follows a specific format. This error signifies that an **invalid or unsupported command** was sent to the device

ERR_DATA_LENGTH_EXCEEDED When writing data to an RFID card, a **specific data length is required**. This error occurs when the length of the data being written is **incompatible with the card's capacity**.

ERR_FLASH_KEY_OPERATION: The **RFIDAX** device uses **KeyA and KeyB** for reading and writing operations. These keys can be stored within the device's memory for future authentication. This error indicates an issue **during the saving or retrieval process of these keys**, which may impact authentication and data access.

ERR_FLASH_ID_RECORD_FAILED: RFIDAX devices can manually store RFID card IDs in memory. These IDs support **reading & write, and update operations**. If an error occurs during these processes, this error is returned to notify the user.

6.4.5 Advanced Errors and System Management

These error codes are related to advanced modes, memory management, communication, protocol validation, and system security. By checking these error codes, users can ensure the proper functioning of the device and take the necessary measures to resolve issues.
These errors typically indicate problems in advanced device settings and system management. Error messages may point to issues at either the hardware or software level. In case of an error, follow the recommended solutions to troubleshoot the problem effectively.

Error Code	Error Description	Possible Solution
0x0064	ERR_ADVANCED_MODE_NOT_SUPPORTED – Command not supported in this mode.	Ensure that the command being used is valid for the selected mode.
0x0068	ERR_CONTACT_MANUFACTURER – Contact manufacturer.	The device may be faulty, require a firmware update, or be operating in the wrong mode. Contact RFIDAX technical support.
0x0070	ERR_INVALID_DEVICE_INFO_COMMAND – Invalid device information command.	Use the correct parameters when sending a device information request command.
0x0071	ERR_RESET_CARD_NOT_READ Reset card not detected..	Ensure the correct reset card is being scanned by the device.
0x0072	ERR_INVALID_DEVICE_ADDRESS nvalid device address.	This error occurs when an incorrect device address is used. Verify and adjust the address if needed.
0x0073	ERR_FLASH_WRITE_FAILURE – Flash memory write failure.	The Flash memory may be damaged or full. Perform a Flash memory cleanup and try again..
0x0074	ERR_INVALID_PROTOCOL – Invalid protocol error.	Ensure that the correct CRC communication protocol is being used.
0x0075	ERR_INVALID_BAUD_RATE – Invalid baud rate error.	The baud rate (communication speed) must be compatible with the device's supported speeds.
0x0076	ERR_INVALID_FORMAT_FLAG – Invalid format flag value.	Check if the flag (status bit) value in the sent command is valid. Ensure the correct format is used.
0x0077	ERR_INVALID_HEX_DATA – Invalid HEX format.	Ensure that the data sent in HEX format is valid. Only use HEX values consisting of characters 0-9 and A-F.
0x0078	ERR_FLASH_ERASE_FAILURE – Flash memory erase failure.	The data in Flash memory may not have been erased. If this error occurs, attempt the erase process again.
0x0079	ERR_UNKNOWN_RESET_FACTORY_CMD – Unknown reset factory command.	This error occurs when an invalid subcommand is sent for a factory reset. Ensure that the correct reset commands and parameters are being used.

Table 14 Advanced Errors and System Management Descriptions

6.4.5.1 Additional Explanation for Advanced Errors and System Management Errors

ERR_ADVANCED_MODE_NOT_SUPPORTED :If this error occurs, check whether the command is valid for the selected mode. If the device is operating in **MODE_BASIC** or **MODE_ENTERPRISE**, note that the command may only be supported in **MODE_ADVANCED**.

ERR_RESET_CARD_NOT_READ: RFIDAX RFIDAX devices feature an advanced communication protocol that supports various configurations. In these complex systems, a **reset tag** is required to restore the device to factory settings. This error indicates an issue during the reading of the reset tag. Ensure that the correct reset card is being scanned, that it is within the **reading area**, and that the **magnetic field is strong enough** for detection.

ERR_INVALID_DEVICE_ADDRESS: This error occurs when an invalid device address is provided during an address change operation. The new device address must be within the valid range of **0 to 255**.

ERR_FLASH_WRITE_FAILURE: This error indicates that the **Flash memory may be damaged or full**. Perform a Flash memory cleanup and retry the operation. If the issue persists, contact the manufacturer for support.ata Flash bellek hasarlı olabilir veya dolu olabilir. **Flash bellek temizleme işlemi yapın ve tekrar deneyin.** Eğer sorun devam ederse üreticiyle iletişime geçin.

ERR_INVALID_PROTOCOL: This error occurs when an incorrect **CRC protocol** is selected. The valid range for **CRC protocol selection** is **0x00 to 0x05** in HEX format. If a value outside this range is sent to the device, this error will be triggered.
ERR_INVALID_BAUD_RATE This error is caused by an incorrect **baud rate selection**. The valid range for **baud rate selection** is **0x00 to 0x0D** in HEX format. If a value outside this range is sent to the device, this error will occur.

ERR_INVALID_FORMAT_FLAG, Format The format flag value must be 0x00 or 0x01 in HEX format. If a different value is received, this error will be returned.

6.5 Device System Monitoring Loop

This section explains how the **system monitoring mechanism** of **RFIDAX** devices operates. The device continuously scans for **RFID cards** and performs predefined actions. Processes such as **card identification, factory reset, buzzer and LED notifications** occur within this system monitoring loop.

6.5.1 Operating Principle of the System Monitoring Loop

RFIDAX devices continuously perform **card detection and system status monitoring**. The device automatically manages **LED, buzzer, and memory operations** when a card is **scanned** or **removed**.
The **system monitoring loop** performs the following operations:

1. Continuously checks whether a card is present.
2. Verifies if the same card has been scanned before to **prevent redundant reads**.
3. If the scanned card's **ID matches the stored Format ID**, the device resets to **factory settings**.
4. If a **new card** is detected or if the same card is scanned after **500ms**, the buzzer and LED **provide a notification**, and the device sends an **automatic message** to the master controller.
5. If the card remains on the reader, the **LED keeps blinking**.
   In **Revision 1** devices, the **LED colors are red and green**.
   In **Revision 2** devices, the **LED colors are red and blue**.
6. When the card is removed, the **LED and buzzer are reset**, and the device returns to **normal operation mode**.
7. If an external **command from the master device** is received, the system **responds instantly** without any data loss.
8. At the end of the monitoring cycle, the device **automatically sends a system monitoring message** to the **master device**.
   For more details on this response, refer to Section 6.6: Automatic System Monitoring Message Sent to the Master Device When a Card is Scanned.

This mechanism ensures that the device **operates continuously** and **responds in real-time** to changes in the environment.

6.5.2 Card Detection and Comparison Process

The device continuously scans for RFID cards and executes the following steps:
When a card is detected, its **ID is compared with the last scanned card ID**.
If a **new card** is detected, the **buzzer is activated**, and an **LED notification** is triggered.
The **card ID** is then compared with the **ID stored in Flash memory**.
If the **stored Format ID matches the scanned ID**, the device **resets to factory settings**.

Card Detection Scenarios:

If the scanned card is **new or different** from the previous one, the **buzzer is activated**, and the **LED starts blinking**. The **device automatically sends a system monitoring message** to the master device.
If the **card ID matches the stored Format ID**, the **device resets to factory settings**.
When the **card is removed**, the **LED and buzzer are reset**, and the device **enters standby mode**.

6.5.2.1 LED and Buzzer Notifications

The usage of **LED and buzzer** is managed as follows:

Status	Buzzer	LED
New card scanned	1 beep	LED starts blinking
Card remains on the device	Silent	LED continues blinking
Card ID matches the stored Format ID	Silent	Device resets to factory settings
Card removed	Silent	LED stays on briefly, then turns off

Table 15 LED and Buzzer NotificationsThese signals provide **real-time feedback** to the user about the device's status.

6.5.3 Actions Taken When a Card is Removed

The **LED stops blinking**.
The **LED stays on briefly** to indicate that the card is no longer present, then turns off.
The **buzzer remains silent**.

This process signals that the device is **ready to detect a new card** and can process another scan.

6.5.4 Working with Flash Memory

The device **automatically manages certain processes** by reading data from Flash memory:

- If the **scanned card ID matches the stored Format ID**, the device **initiates a factory reset**.
- If the **card ID does not match the stored data**, the card is **treated as a new card**, and standard processing continues.

This mechanism allows the device to be **managed using predefined special cards** (e.g., using a designated card to reset the device).

6.5.5 Importance of the System Monitoring Loop

Enables the **automatic detection and processing** of cards.
Compares the scanned ID with stored data, allowing factory reset functionality.
Provides **real-time user feedback** via LED and buzzer signals.
Ensures that the device **continuously operates** and responds to changes in real-time.
This system allows **continuous RFID card monitoring**, ensuring the device provides **relevant notifications** to the user at all times.

6.6 Automatic Response from the Device When a Card is Scanned (System Monitoring Message Automatically Sent to the Master Device)

This section provides a detailed explanation of the **automatic response message** that the **RFIDAX** device sends when an **RFID card is scanned**. This message is automatically transmitted to the **master device** (e.g., a computer, control unit) and contains **card information and the current status of the device**.

Message size

- When **CRC is active** (DATA_INTEGRITY_CHECK), the message size is **32 bytes**.
- When **CRC is disabled** (NO_DATA_INTEGRITY_CHECK), the message size is **30 bytes**.

6.6.1 Definitions and Data Structure

The purpose of this message is to **confirm the successful scanning of a card** and to **share the device's current configuration**. The message consists of the following components:

- **Start Byte (0xAA):** Indicates the beginning of the message. It is structured using hexadecimal values. This value is **always fixed**. Applicable to **RFIDAX devices operating at 13.56 MHz**.**Offset number: 0**.
- **Device Address:** Allows the master device to communicate with a specific RFID reader. The device address ranges between 1 and 255.

Factory default address is 01, but it can be modified by the user if needed Offset number: 1**.**

- **Message Type**: Indicates that this is a system monitoring message.
  The system monitoring message type is always 0x01.
  This value is fixed. *(Refer to Section 6.7.2.1 Message Type Information for further details.)*
  Offset number: 2.
- **Hardware and Software Versions**: Contains information about the device's current hardware and software versions.
  This information is important for firmware updates, device capabilities, and diagnostics, providing valuable insights for both users and manufacturers.
  Offset range: 3-10.

Relevant offset details:

Offset	Information
3-4	Hardware Version (Major)
5-6	Hardware Version (Minor)
7-8	Firmware Version (Major)
9-10	Firmware Version (Minor)

These values change as the device undergoes firmware and hardware updates.

- **Build Number and Date**: Facilitates firmware version tracking.
  Provides details about the latest software build and update history.
  Offset range: 11-20.

Relevant offset details:

Offset	Information
11-14	Build Number
15-20	Build Date

(These values evolve as the device software is updated.)

- **Communication Mode** Specifies the **communication interface** used by the device (USB, RS485, Type-C, etc.).
  Offset number: 21
  The **hex value at this offset** represents the device's communication mode.

Communication Type	Hex Value
COMM_USB	0x00
COMM_RS485	0x01
COMM_TYPE_C	0x02
COMM_MOBILE	0x03
COMM_ETHERNET	0x04
COMM_WIFI	0x05

Table 16 Communication Types

- **CRC Modu**: Defines the **error detection mechanism** that verifies data integrity.
  Offset number: 22.The **hex value at this offset** indicates whether CRC verification is enabled or disabled.

Data Integrity Mode	Hex Value
NO_DATA_INTEGRITY_CHECK	0x00
DATA_INTEGRITY_CHECK	0x01

Table 17 Data Integrity Modes

- **Operasyon Modu**: Specifies the **device's working mode** (Basic, Advanced, Enterprise, etc.).
  Offset number: 23. The **hex value at this offset** represents the current operation mode of the device.

Operation Mode	Hex Value
MODE_BASIC	0x00
MODE_ADVANCED	0x01
MODE_ENTERPRISE	0x02
MODE_CUSTOM	0x03

Table 18 Operation Modes

- **Reserved Byte (Offset 24):**
  This byte is **reserved for future use**.
- **Card Identification Number (UID):** The unique identification number of the scanned card.
  Provides the **unique ID information** of the scanned RFID card. Offset range: 25-28.
- **Reserved Byte (Offset 29):**
  This byte is **reserved** for the manufacturer.
- **CRC Check Value:** Used as an **error control byte** to verify **data integrity**.
  Computes the **CRC for the received message**. The factory default CRC calculation method is **CRC-16 CCITT**. Offset range: 30-31.

6.6.2 Byte-by-Byte Explanation

The table below provides a detailed breakdown of the **response message**, explaining the purpose and meaning of each byte in the message structure:

Byte Number	Byte Value (Hex)	Byte Description	General Explanation
0 byte	0xAA	Başlangıç Baytı	Always fixed
1 byte	Variable	Cihaz Adresi	Can vary depending on the configuration
2 byte	0x01	Mesaj Türü	System monitoring message type (Always fixed)
3 byte	Variable	Hardware Version (Major)	2 bytes allocated for major version information (3-4 bytes)
4 byte	Variable	Hardware Version (Major)	
5 byte	Variable	Hardware Version (Minor)	2 bytes allocated for minor version information (5-6 bytes)
6 byte	Variable	Hardware Version (Minor)	
7 byte	Variable	Software Version (Major)	2 bytes allocated for major version information (7-8 bytes)
8 byte	Variable	Software Version (Major)	
9 byte	Variable	Software Version (Minor)	2 bytes allocated for minor version information (9-10 bytes)
10 byte	Variable	Software Version (Minor)	
11 byte	Variable	Build Number	4 bytes allocated for build number information (11-14 bytes)
12 byte	Variable		
13 byte	Variable		
14 byte	Variable		
15 byte	Variable	Build Date	6 bytes allocated for build date information (15-20 bytes)
16 byte	Variable	Build Date	
17 byte	Variable	Build Date	
18 byte	Variable	Build Date	
19 byte	Variable	Build Date	
20 byte	Variable	Build Date	
21 byte	Variable	Communication Mode	1 byte allocated for communication mode
22 byte	Variable	CRC Mode	1 byte allocated for operation mode
23 byte	Variable	Operation Mode	1 byte allocated for operation mode
24 byte	Variable	Reserved Field	Reserved for future development
25 byte	Variable	Card ID Value	4 bytes allocated for card ID value
26 byte	Variable	Card ID Value	
27 byte	Variable	Card ID Value	
28 byte	Variable	Card ID Value	
29 byte	Variable	Reserved Byte	Do not use (Reserved for card manufacturer-specific data)
30 byte	Variable	CRC Check Value	2-byte control value calculated using CRC-16 CCITT (30-31 bytes)
31 byte	Variable	CRC Check Value	

Table 19 Byte-by-Byte Explanation of the System Monitoring Message Structure

6.6.2.1 Summary of Byte-by-Byte Explanation

The **summary of the System Monitoring Message** provided above is presented in the following table:

Ofset (Byte)	Hex Value	Byte Description	Detailed Description
0	0xAA	Start Byte	Start byte of the data packet, always fixed.
1	Variable	Device Address	A variable byte indicating the address of the communicating device.
2	0x01	Message Type	Fixed value for system monitoring messages.
3-4	Variable	Hardware Version (Major)	2-byte field representing the **major** hardware version
5-6	Variable	Hardware Version (Minor)	2-byte field representing the **minor** hardware version.
7-8	Variable	Software Version (Major)	2-byte field representing the **major** software version.
9-10	Variable	Software Version (Minor))	2-byte field representing the **minor** software version.
11-14	Variable	Build Number	4-byte field containing the software build number
15-20	Variable	Build Date	6-byte field containing the software build date.
21	Variable	Communication Mode	Byte indicating the communication type (e.g., USB, RS485, etc.).
22	Variable	CRC Mode	Byte indicating the error control mode (**'0x01': CRC enabled**).
23	Variable	Operation Mode	Byte specifying the system operation mode (e.g., Basic, Advanced, etc.).
24	Variable	Reserved Field	Reserved for future enhancements.
25-28	Variable	Card ID Value	4-byte field reserved for card identification data.
29	Variable	Reserved Field	Reserved for manufacturer use, do not modify.
30-31	Variable	CRC Check Value	2-byte control value calculated using **CRC-16 CCITT**.

Table 20 Byte-by-Byte Explanation Summary of the System Monitoring Structure

6.6.2.1.1 Example System Monitoring Message Explanation

This section details the **automatic response message** sent by the **RFIDAX** device to the master device when an **RFID card is scanned**. The example message below includes **system status information** and details of the scanned card.

Example Message - 1
AA 01 01 00 01 00 02 00 02 00 00 00 00 00 25 32 30 32 34 31 32 00 01 01 00 C3 C7 46 FC BE 24 BA

Byte-by-Byte Explanation of the Message Structure

Offset (Byte)	Hex Value	Byte Description	Description
0	0xAA	Start Byte	Indicates the start of the message. Always fixed.
1	0x01	Device Address	Specifies the address of the device.
2	0x01	Message Type	Identifies the message as a system monitoring message.
3-4	0x00 0x01	Hardware Version (Major)	Device's main hardware version (v1.x).
5-6	0x00 0x02	Hardware Version (Minor)	Device's minor hardware version (v1.2).
7-8	0x00 0x02	Software Version (Major)	Software version (v2.x).
9-10	0x00 0x00	Software Version (Minor)	Software sub-version (v2.0).
11-14	0x00 0x00 0x00 0x25	Build Number	Software build number (37).
15-20	0x32 0x30 0x32 0x34 0x31 0x32	Build Date	Software release date (2024-12).
21	0x00	Communication Mode	0x00: USB communication mode (COMM_USB).
22	0x01	CRC Mode	0x01: CRC enabled, data integrity check is active (DATA_INTEGRITY_CHECK).
23	0x01	Operation Mode	0x01: Advanced mode is active (MODE_ADVANCED).
24	0x00	Reserved Field	Reserved for future use.
25-28	0xC3 0xC7 0x46 0xFC	Card ID	The unique identification number of the scanned RFID card.
29	0xBE	Reserved Field	Reserved for the manufacturer, not to be used.
30-31	0x24 0xBA	CRC Check Value	2-byte error control value calculated using CRC-16 CCITT.

Table 21 Example System Monitoring Message Structure
(Captured from an actual RFIDAX device and scanned card)

Summary

This example message illustrates the automatic response sent by the device when an RFID card is scanned.

- The total message length is 32 bytes because CRC integrity check is enabled.
- If CRC were disabled, the message length would be 30 bytes, and the last two CRC bytes (0x24 0xBA) would be omitted.

This structured message allows the system monitor to send an up-to-date status report and scanned card details to the master device. Users can analyze this message to obtain real-time device status and card information.

Example Message - 2
AA 01 01 00 01 00 02 00 02 00 00 00 00 00 25 32 30 32 34 31 32 02 01 00 00 C3 C7 46 FC BE 5A BC

Offset (Byte)	Hex Value	Byte Description	Description
0	0xAA	Start Byte	The start byte of the data packet. Always fixed.
1	0x01	Device Address	The assigned variable address of the device (0x01 for device 1).
2	0x01	Message Type	Indicates that this is a system monitoring message.
3-4	0x00 0x01	Hardware Version (Major)	Device's main hardware version (v1.x).
5-6	0x00 0x02	Hardware Version (Minor)	Device's minor hardware version (v1.2).
7-8	0x00 0x02	Software Version (Major)	Software version (v2.x).
9-10	0x00 0x00	Software Version (Minor)	Software sub-version (v2.0).
11-14	0x00 0x00 0x00 0x25	Build Number	The software build number (37).
15-20	0x32 0x30 0x32 0x34 0x31 0x32	Build Date	The date when the software was built (2024-12).
21	0x02	Communication Mode	0x02: Type-C communication mode (COMM_TYPE_C).
22	0x01	CRC Mode	0x01: CRC is active, data integrity check enabled (DATA_INTEGRITY_CHECK).
23	0x00	Operation Mode	0x00: Basic mode is active (MODE_BASIC).
24	0x00	Reserved Field	Reserved for future use.
25-28	0xC3 0xC7 0x46 0xFC	Card ID	The unique identification number of the scanned RFID card.
29	0xBE	**Reserved Field**	Reserved by the manufacturer, do not use.
30-31	0x5A 0xBC	**CRC Check Value**	The CRC-16 CCITT error control value.

Table 22 Example of the System Monitoring Structure (Data captured from an actual RFIDAX device and scanned card).

The table below summarizes the key details from the **system monitoring message** received from the device:

Kategori	Değer
Communication Mode	Type-C
Data Integrity Check	Enabled (CRC Active)
Operation Mode	Basic
Hardware Version	1.2
Software Version	2.0
Build Number	37
Build Date	2024-12
Card ID	C3C746FC
CRC	5ABC

These details indicate the **communication type, software/hardware versions, and scanned card ID**.The **message length is 32 bytes** since **CRC verification is enabled**.
If **CRC were disabled**, the message length would be **30 bytes**, omitting the **last two CRC bytes (0x5A 0xBC)**.

Example Message - 3
AA 03 01 00 01 00 02 00 03 00 02 00 00 00 4B 32 30 32 35 30 31 03 01 02 00 66 A7 7B DA 60 F8 84

Offset (Byte)	Hex Value	Field Name	Detailed Description
0	0xAA	Start Byte	Fixed value indicating the start of the data packet.
1	0x03	Device Address	The assigned address of the device (variable).
2	0x01	Message Type	Fixed value for system monitoring messages.
3-4	0x00 0x01	Hardware Version (Major)	2-byte field indicating the major hardware version.
5-6	0x00 0x02	Hardware Version (Minor)	2-byte field indicating the minor hardware version.
7-8	0x00 0x03	Software Version (Major)	2-byte field indicating the major software version.
9-10	0x00 0x02	Software Version (Minor)	2-byte field indicating the minor software version.
11-14	0x00 0x00 0x00 0x4B	Build Number	4-byte field indicating the software build number (75).
15-20	0x32 0x30 0x32 0x35 0x30 0x31	Build Date	6-byte field indicating the software build date (2025-01).
21	0x03	Communication Mode	0x03: Mobile communication mode selected (COMM_MOBILE).
22	0x01	CRC Mode	0x01: CRC enabled, data integrity check active (DATA_INTEGRITY_CHECK).
23	0x02	Operation Mode	0x02: Enterprise mode active (MODE_ENTERPRISE).
24	0x00	Reserved Field	Fixed value 0x00, reserved for future enhancements.
25-28	0x66 0xA7 0x7B 0xDA	Card ID	The unique identification number of the scanned RFID card.
29	0x60	Reserved Field	Reserved by the manufacturer, not to be used.
30-31	0xF8 0x84	CRC Check Value	2-byte error control value calculated using CRC-16/MODBUS.

Table 23 Example System Monitoring Message Structure (Captured from an actual RFIDAX device and scanned card.)

The table below summarizes the key details from the **system monitoring message** received from the device:

Category	Value
Communication Mode	Mobile (COMM_MOBILE)
Data Integrity Check	Enabled (CRC Active)
Operation Mode	Enterprise (MODE_ENTERPRISE)
Hardware Version	1.2
Software Version	3.2
Build Number	75
Build Date	2025-01
Card ID	66A77BDA
CRC	F884

6.7 Device Response Message Structure

This section describes the **message structure** of responses sent by **RFIDAX** devices.
It explains how messages sent by the device are structured and how they should be interpreted. The device sends two different types of messages:

Status Messages (report_status)

A message type that contains **the device's operational status, error codes, or success information**.
Fixed length of 6 bytes.
Reports errors or success states based on **the communication type** and **data integrity mode.**
If requested (**Custom Mode**), error messages can be sent with detailed explanations.

Data Packets (send_data_packet)

A message type that transmits **RFID card data, device-specific information, configuration settings, and stored memory data**.
Sends responses based on the **command received**.
Variable length.
The **message format** depends on the **communication type** and **data integrity mode**.

These two message types are explained in detail in the following sections.

6.7.1 Status Messages (report_status)

This section has been prepared to interpret **status messages** sent by the device.
When specific events or error conditions occur, the **report_status()** function generates a **6-byte message** and transmits it to the target device.
The message contains:

Header (HEADER)
Device Address (DEVICE_ADDRESS)
Status Code (STATUS_CODE_HIGH, STATUS_CODE_LOW)
CRC information (optional)

If **CRC is enabled**, the **last two bytes** of the message are used to ensure **data integrity**.
Additionally, when **with_description** is enabled, a text containing the meaning of the error code can also be included.
This document does not explain the with_description feature.
This function generates the **message format** used to report the device status, the status of operations performed, or error codes. The message follows a **specific structure** to convey the **device's error or status information** to the target device.

HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)

The **header** is always **fixed** for report_status messages and has a **hex value of 0xBB**.
The **device address** can be modified; for factory-default devices, the **hex value is 0x01**.
STATUS_CODE_HIGH and **STATUS_CODE_LOW** values change depending on the reported condition.
For the **full list of status codes**, refer to Section 6.4: Device Status Messages, Error Codes, and Descriptions.
If **CRC is enabled**, the **CRC values of the corresponding communication protocol** are appended at the end.

Additionally, when the **with_description** parameter is enabled, a **text explanation** of the error code can be included.
This structure allows the **device's operational state and potential errors** to be easily analyzed.

Status Messages (Report_status) Example 1

Byte	Content	Description	Example Value
0	HEADER	Message start byte	0XBB
1	DEVICE_ADDRESS	Device address	0x01
2	STATUS_CODE_HIGH	High byte of the status code	0x00
3	STATUS_CODE_LOW	Low byte of the status code	0x00
4	CRC_HIGH (opsiyonel)	CRC high byte (for data integrity)	0x5C
5	CRC_LOW (opsiyonel)	CRC low byte (for data integrity)	0x3E

Table 24 (Captured from an actual RFIDAX device and scanned card)

When BB 01 00 00 5C 3E is examined, the 2nd Byte 0x00 and 3rd byte 0x00 value corresponds to “0x0000 SUCCESS – Operation successful.” Please review the relevant section. (6.4.1 General System Status and Errors)

Status Messages (Report_status) Example 2

To better understand status messages, an **invalid LED command** was intentionally sent.

Sent Command:

Command Sent						
0xAA	0x01	0x01	0x01	0x03	0x3F	0x10

Received Message:

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x10	0x4E	0x0F

When BB 01 00 10 4E 0F is examined, the 2nd Byte 0x00 and 3rd byte 0x10 values correspond to “ERR_INVALID_LED_STATE – Invalid LED status. Check whether the LED is on or off correctly.” Please review the relevant section. (6.4.3 Errors Related to LEDs and Indicators)

Status Messages (Report_status) Example 3

To better understand status messages, a **command to activate the buzzer for 100ms** was sent.

Sent Command:

Command Sent						
0xAA	0x01	0x06	0x00	0x0A	0x18	0x98

Received Message:

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

When BB 01 00 BB 01 00 00 5C 3E is examined, the 2nd Byte 0x00 and 3rd byte 0x00 value corresponds to "0x0000 SUCCESS – Operation successful." Please review the relevant section. (6.4.1 General System Status and Errors)

The buzzer returned a successful message for the 100ms activation request and the buzzer opened for 100ms.

Status Messages (Report_status) Example 4

To better understand status messages, a **request was sent to read 2 blocks from an RFID card**.

However, **no card was placed in the antenna reading area**.

Sent Command:

Command Sent								
0xAA	0x01	0x08	0x01	0x02	0x02	0x02	0xDE	0xF6

The message received is below,

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x20	0x78	0x5C

When BB 01 00 10 4E 0F is examined, the 2nd Byte 0x00 and 3rd byte 0x20 values correspond to "ERR_CARD_NOT_FOUND – Card not found. Check that the card is properly placed in the reader area." Please review the relevant section. (6.4.4 RFID Card Reading/Writing Errors)

Status Messages (Report_status) Example 5

To better understand status messages, a **request was sent to read 2 blocks from an RFID card**.

The difference from the **previous example** is that this example demonstrates the response when **CRC is disabled** on the device (**[Data Integrity Check]: Disabled (No CRC)**).

However, **no card was placed in the antenna reading area**.

Sent Command:

Command Sent								
0xAA	0x01	0x08	0x01	0x02	0x02	0x02	0xDE	0xF6

The message received is below,

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x20	0x00	0x00

When BB 01 00 20 4E 0F is examined, the 2nd Byte 0x00 and 3rd byte 0x20 values correspond to "ERR_CARD_NOT_FOUND – Card not found. Check that the card is properly placed in the reader area." Please review the relevant section. (6.4.4 RFID Card Reading/Writing Errors) If you pay attention, CRC_HIGH and CRC_LOW bytes 0x00 has come, this indicates that the Data Integrity Mode is NO_DATA_INTEGRITY_CHECK.

6.7.2 Data Packets (send_data_packet)

This section is prepared to interpret the data packets sent by the device. The device, after certain operations or in response to a request, generates a dynamically sized message using the **send_data_packet()** function and transmits it to the target device. The message includes a header (**HEADER**), device address (**DEVICE_ADDRESS**), message type (**MESSAGE_TYPE**), data (**DATA**), and optionally CRC information. If CRC is enabled, the last two bytes of the message are used to ensure data integrity. This structure allows the data sent by the device to be processed in a standard format and received correctly by the target device.

HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xAA	0x00-0xFF 1 Byte	Variable (1 Byte)	Variable Length	Variable (1 Byte)	Variable (1 Byte)

This structure defines the standard format of the data packets sent by the device and explains the different components of the message.

HEADER (0xAA): This is the message start byte and is always fixed. When the device sends data using the **send_data_packet** function, the first byte of the message will always be **0xAA**.

DEVICE_ADDRESS (0x01 - 255 Variable): This is the device address, which is set to **0x01** in factory default devices. It can be modified as needed.

MESSAGE_TYPE (Variable Values): Indicates the type of function responding to the message. Each function has its own **Message Type** value. Refer to Section 6.7.2.1 Message Type Information for these values.

DATA (Variable Length Between 5 Bytes - 1024 Bytes): Contains the sent data content. The length can vary between **5 bytes and 1024 bytes**.

CRC_HIGH & CRC_LOW (Optional): The **STATUS_CODE_HIGH** and **STATUS_CODE_LOW** values change depending on the received status. A separate **CRC** is calculated for each message. If **CRC is enabled**, it is calculated according to the communication protocol used and added to the end of the message.

6.7.2.1 Message Type Information

The **Message Type** is a **one-byte value** in the **send_data_packet** structure. The purpose of this information is to define the **type and function** of the response sent by the **RFIDAX** device.

The **Message Type** determines which operation and function the response corresponds to.

- It consists of **1 byte** of data.
- Interpreting the **Message Type** from the **send_data_packet** makes it significantly easier to understand the received message.

Below are the **Message Type values** and their descriptions.

The message types received from the device are **critical** for understanding the system's operation and managing the data correctly. Each **Message Type** represents a **specific operation or data type** and forms the **core structure** that organizes the device's functionality.

Message Code	Message Type	Description
0x01	System_Monitor	Message type used to send system monitoring data.
0x02	Device_Version_Info	Message type used to send all version-related data of the device.

Table 25 Message Type Information

6.7.2.1.1 RFID Card Reading Operations Messages and Message Type Information

The device transmits critical information such as UID, SAK, ATQA of the read RFID cards through these message types. At the same time, card-specific keys (KeyA, KeyB) and optional security keys can also be shared with messages.

Message Code	Message Type	Description
0x07	Read_Card_UID	Message type containing the UID (Unique Identifier) of the scanned RFID card.
0x08	Read_Card_ATQA	Message type containing the ATQA value of the scanned RFID card.
0x09	Read_Card_SAK	Message type containing the SAK value of the scanned RFID card.
0x0A	Read_Card_Info	Message type sending **UID, SAK, and ATQA values** of the scanned RFID card in a single message.
0x0B	Display_Key_A	Message type displaying **KeyA** for RFID cards (**0x11 is the same, used for data integrity**).
0x0C	Display_Key_B	Message type displaying **KeyB** for RFID cards (**0x12 is the same, used for data integrity**).
0x0E	Display_Key_Optional_KeyA	Message type sending **Optional KeyA** for RFID cards.
0x0F	Display_Key_Optional_KeyB	Message type sending **Optional KeyB** for RFID cards.
0x10	Blok_Verilerini_Okuma	Message type used to read a **specific block** of an RFID card.
0x11	KeyA	Message type containing the **KeyA value** of an RFID card.
0x12	KeyB	Message type containing the **KeyB value** of an RFID card.

Table 26 RFID Card Reading Message Types and Message Type Information

6.7.2.1.2 Flash Memory ID Update Messages and Message Type Information

CThe **identity and configuration data** stored in the device's **internal memory** contain **critical information** that needs to be updated periodically. These message types are used to **update specific IDs** stored in **flash memory**.

Message Code	Message Type	Description
0x20	Flash_Update_ID_0	Message type used to update **ID 0** in flash memory.
0x21	Flash_Update_ID_1	Message type used to update **ID 1** in flash memory.
0x22	Flash_Update_ID_2	Message type used to update **ID 2** in flash memory.
0x23	Flash_Update_ID_3	Message type used to update **ID 3** in flash memory.
0x24	Flash_Update_ID_4	Message type used to update **ID 4** in flash memory.
0x25	Flash_Update_ID_5	Message type used to update **ID 5** in flash memory.
0x26	Flash_Update_ID_6	Message type used to update **ID 6** in flash memory.
0x27	Flash_Update_ID_7	Message type used to update **ID 7** in flash memory.
0x28	Flash_Update_ID_8	Message type used to update **ID 8** in flash memory.
0x29	Flash_Update_ID_9	Message type used to update **ID 9** in flash memory.

Table 27 Flash Memory Update Messages and Message Type Information

6.7.2.1.3 Flash Memory ID Read Messages and Message Type Information

The **identity and configuration data** stored in the **device's internal memory** contain **critical information** that needs to be accessed periodically. These message types are used to **read specific IDs** stored in **flash memory**.

Message Code	Message Type	Description
0x50	Flash_Read_ID_0	Message type used to read **ID 0** from flash memory.
0x51	Flash_Read_ID_1	Message type used to read **ID 1** from flash memory.
0x52	Flash_Read_ID_2	Message type used to read **ID 2** from flash memory.
0x53	Flash_Read_ID_3	Message type used to read **ID 3** from flash memory.
0x54	Flash_Read_ID_4	Message type used to read **ID 4** from flash memory.
0x55	Flash_Read_ID_5	Message type used to read **ID 5** from flash memory.
0x56	Flash_Read_ID_6	Message type used to read **ID 6** from flash memory.
0x57	Flash_Read_ID_7	Message type used to read **ID 7** from flash memory.
0x58	Flash_Read_ID_8	Message type used to read **ID 8** from flash memory.
0x59	Flash_Read_ID_9	Message type used to read **ID 9** from flash memory.

Table 28 Flash Memory Read Messages and Message Type Information

6.7.2.1.4 Flash Memory Reset Messages and Message Type Information

Sometimes, it is necessary to **clear old data** stored in the device's memory or **reset specific identity information**. These message types are used to **reset specific IDs** in **flash memory**.

Message Code	Message Type	Description
0x8C	Flash_Reset_ID_0	Message type used to reset **ID 0** in flash memory.
0x8D	Flash_Reset_ID_1	Message type used to reset **ID 1** in flash memory.
0x8E	Flash_Reset_ID_2	Message type used to reset **ID 2** in flash memory.
0x8F	Flash_Reset_ID_3	Message type used to reset **ID 3** in flash memory.
0x90	Flash_Reset_ID_4	Message type used to reset **ID 4** in flash memory.
0x91	Flash_Reset_ID_5	Message type used to reset **ID 5** in flash memory.
0x92	Flash_Reset_ID_6	Message type used to reset **ID 6** in flash memory.
0x93	Flash_Reset_ID_7	Message type used to reset **ID 7** in flash memory.
0x94	Flash_Reset_ID_8	Message type used to reset **ID 8** in flash memory.
0x95	Flash_Reset_ID_9	Message type used to reset **ID 9** in flash memory.
0xC8	Flash_Display_All_IDs	Message type used to display **all IDs** stored in flash memory.

Table 29 Flash Memory Reset Messages and Message Type Information

After an ID is reset, the system **automatically replaces it with the value 0xFF**.

.

6.7.2.1.5 Format ID Read Message and Message Type Information

The **format identification information** of the device plays a **critical role** in **data processing operations**. This message is used to **read and verify the identity number** of the **Format ID**.

Message Code	Message Type	Description
0x60	ReadFlash_Format_ID_DATA	Message type used to **read the identity number** of the **Format ID**.

Table 30 Format ID Read Messages and Message Type Information

Data Packets (send_data_packet) Example 1

To better understand the **send_data_packet**, an **update operation for ID number 2** will be performed.

Sent Command:

Command Sent											
0xAA	0x01	0x0C	0x01	0x04	0x02	0xAA	0x22	0xCC	0xDD	0x8B	0x84

Received Message:

Send Data Packet Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xAA	0x01	0x22	0xAA 0x22 0xCC 0xDD	0xED	0x8A

When AA 01 22 AA 22 CC DD ED 8A is examined, 2 byte information is Message Type Information, its value is 0x22.

When we look at its equivalent from the Message Type Information table (6.7.2.1.2 Flash Memory ID Update Messages and Message Type Information), it gives the information "Flash_Update_ID_2 Message type used to update ID 2 in Flash memory."

3,4,5,6. Byte information is the data message, it gives the updated ID information. This information is 0xAA,0x22,0xCC,0xDD in order.

If you pay attention, CRC_HIGH and CRC_LOW byte information is 0Xed and 0x8A, this shows that the Data Integrity Mode is DATA_INTEGRITY_CHECK.

Data Packets (send_data_packet) Example 2

To better understand the **send_data_packet**, a **UID read (RFID card ID read) operation** will be performed.

Sent Command:

Command Sent						
0xAA	0x01	0x07	0x01	0xFF	0xA3	0x23

Received Message:

Send Data Packet Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xAA	0x01	0x07	0x66 0xA7 0x7B 0xDA	0x1D	0x28

When AA 01 07 66 A7 7B DA 1D 28 is examined, the 2nd byte represents the Message Type, with a value of 0x07.
Referring to the Message Type Table (6.7.2.1.1 RFID Card Reading Message Types and Message Type Information), 0x07 corresponds to "Read_Card_UID – Message type containing the UID of the scanned RFID card."The 3rd, 4th, 5th, and 6th bytes represent the data message, providing the updated ID information. These values are 0x66, 0xA7, 0x7B, 0xDA in sequence.
Notably, the CRC_HIGH and CRC_LOW bytes (0x1D and 0x28) indicate that the Data Integrity Mode is DATA_INTEGRITY_CHECK.

Data Packets (send_data_packet) Example 3

To better understand the **send_data_packet**, a **Device Info query** will be performed.

Sent Command:

Command Sent					
0xAA	0x01	0x0D	0x01	0x57	0x50

Received Message:00 01 00 02 00 03 00 02 00 00 00 4B 32 30 32 35 30 31 03 01 01 00

Send Data Packet Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xAA	0x01	0x02	0x00 0x01 0x00 0x02 0x00 0x03 0x00 0x02 0x00 0x00 0x00 0x4B 0x32 0x30 0x32 0x35 0x30 0x31 0x03 0x01 0x01 0x00	0xF7	0xEC

When AA 01 02 00 01 00 02 00 03 00 02 00 00 00 4B 32 30 32 35 30 31 03 01 01 00 F7 EC is examined, the 2nd byte represents the Message Type, with a value of 0x02.

Referring to the Message Type Table (6.7.2.1 Message Type Information), 0x02 corresponds to "Device_Version_Info – Message type used to send all version-related data of the device."

The 3rd to 24th bytes contain the version message data. For a detailed analysis of this data structure, please refer to the relevant section..

Data Packets (send_data_packet) Example 4

To better understand the **send_data_packet**, a **Device Format ID query** will be performed.

Sent Command:

Command Sent					
0xAA	0x01	0x0F	0x02	0x01	0x51

Received Message:

Send Data Packet Response								
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA				CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xAA	0x01	0x60	0x93	0x32	0xEF	0xF6	0xE2	0x20

When **AA 01 60 93 32 EF F6 E2 20** is examined, the **2nd byte** represents the **Message Type**, with a value of **0x60**.

Referring to the **Format ID Read Message Table (6.7.2.1.5 Format ID Read Message and Message Type Information)**, **0x60** corresponds to **"ReadFlash_Format_ID_DATA – Message type used to read the identity number of the Format ID."**

The **3rd, 4th, 5th, and 6th bytes (0x93, 0x32, 0xEF, 0xF6)** contain the **ID information that formats the system**.

Notably, the **CRC_HIGH and CRC_LOW bytes (0xE2 and 0x20)** indicate the **CRC integrity of the response**.

7.0 Device Commands Guide

This guide explains the **command sets** supported by the device, including the **function, purpose, and operation** of each command. Commands allow the device to be controlled, facilitate data exchange, and execute specific operations.

Device commands start with a **HEADER** and include the **device address, command type, parameters, and optionally CRC information**. These commands are **dynamic in length** and are used to control the target **RFIDAX** device.

HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)

Section Descriptions

Section	Description
HEADER	Determines the **beginning of the command**. It takes a fixed value and is found at the beginning of all messages.
DEVICE_ADDRESS	Specifies **which device** the command is sent to. It contains the device's address.
COMMAND_TYPE	The key code that determines the **main type of the command**. For example, different command types such as **LED control, RFID reading, and memory operations** are represented.
SUB_TYPE (If applicable)	Used in some commands to determine the **sub-operation type**. For example, in RFID commands, different sub-commands such as **reading UID, SAK, or ATQA** are indicated.
DATA (If applicable)	Contains additional data depending on the command's content. For example, if data is to be written to a specific block on an RFID card, it is included here.
CRC (Optional)	The **error-checking mechanism** used to ensure data integrity. If enabled, it is appended at the end of the message and used for validation.

Table 31 Command Components and Descriptions

The Command_Type information will be explained sequentially in the description of the relevant commands.

7.1 Command Categories and Descriptions

The device's commands are divided into different categories:

- **System Control Commands** (Used for general device control and management.)
- **LED and Buzzer Commands** (Used for visual and sound notifications.)
- **RFID Card Commands** (Used for reading, writing, and verifying operations on RFID cards.)
- **Memory Management Commands** (Used for reading, writing, and resetting flash memory.)
- **Device Configuration Commands** (Used for changing system settings such as baud rate, communication protocol, and device ID.)
- **Factory Reset Commands** (Used to reset the device and restore factory settings.)

The device consists of a total of **31 commands and sub-commands**.

7.1.1 Command Categories and Subcategories Explanations Based on Operating Modes

This section explains how the commands that can be sent to the **RFIDAX** device are supported in different **operating modes**. The device functionality can be configured in three different modes: **MODE_BASIC, MODE_ENTERPRISE, and MODE_ADVANCED**. These modes are categorized into **system control, LED and buzzer management, RFID card operations, memory management, device configuration, and factory reset commands**.
The table below shows which commands are **active** in which modes, allowing users to clearly understand which features they can utilize based on their **selected operating mode**. **MODE_BASIC** provides an **entry-level usage** with fundamental functions, while **MODE_ENTERPRISE** offers **enhanced security and management options**, allowing for more comprehensive operations. **MODE_ADVANCED** unlocks **all capabilities** of the device, providing **maximum customization and management flexibility** as the most comprehensive mode.

With this structure, users can select the most suitable operating mode according to their **needs and project requirements**, ensuring they get the **most efficient use** of the device. The following table presents a **detailed overview** of how each **command category** functions in different modes.

Categories and Command Codes	Mode Information			Description
	MODE_BASIC	MODE_ENTERPRISE	MODE_ADVANCED	
1- System Control Commands				Get device version information
0x0A	❌ (Not Active)	✅ (Active)	✅ (Active)	Software reset command
0x0D	✅ (Active)	✅ (Active)	✅ (Active)	
2- LED and Buzzer Control Commands				
0x01	✅ (Active)	✅ (Active)	✅ (Active)	Turn a specific LED on/off
0x02	❌ (Not Active)	✅ (Active)	✅ (Active)	Toggle a specific LED (blink)
0x03	❌ (Not Active)	✅ (Active)	✅ (Active)	Toggle two LEDs sequentially (blink alternately)
0x04	❌ (Not Active)	❌ (Not Active)	✅ (Active)	Turn off all LEDs
0x05	✅ (Active)	✅ (Active)	✅ (Active)	Turn buzzer on/off
0x06	❌ (Not Active)	❌ (Not Active)	✅ (Active)	Activate buzzer for a specific duration
3- RFID Card Operation Commands				
0x07 - RFID card recognition, card reading and writing command				
3-a) 0x07 / 0x01	✅ (Active)	✅ (Active)	✅ (Active)	Read Card ID (UID)
3-b) 0x07 / 0x02	❌ (Not Active)	❌ (Not Active)	✅ (Active)	Read SAK value
3-c) 0x07 / 0x03	❌ (Not Active)	❌ (Not Active)	✅ (Active)	Read ATQA value
3-d) 0x07 / 0x04	❌ (Not Active)	❌ (Not Active)	✅ (Active)	Read full card info (UID + SAK + ATQA)
3-e) 0x07 / 0x05	❌ (Not Active)	✅ (Active)	✅ (Active)	Display stored security keys
0x08	❌ (Not Active)	✅ (Active)	✅ (Active)	RFID block reading command
0x09		❌ (Not Active)	✅ (Active)	RFID block write command

4- Memory Management Commands				
0x0B - Write or read optional KeyA and KeyB values				
4-a) 0x0B / 0x01	(Not Active)	(Active)	(Active)	Write keyA
4-b) 0x0B / 0x02	(Not Active)	(Active)	(Active)	Write keyB
4-c) 0x0B / 0x03	(Not Active)	(Active)	(Active)	Read keyA
4-d) 0x0B / 0x04	(Not Active)	(Active)	(Active)	Read keyB
0x0C - ID operations				
4-e) 0x0C / 0x01				Update an ID
4-f) 0x0C / 0x02	(Not Active)	(Not Active)	(Active)	Read an ID
4-g) 0x0C / 0x03	(Not Active)	(Not Active)	(Active)	Reset an ID
4-h) 0x0C / 0x04	(Not Active)	(Not Active)	(Active)	Display all stored IDs
5- Device Configuration Commands				
0x0E - Device configuration commands				
5-a) 0x0E / 0x01	(Not Active)	(Active)	(Active)	Change Device ID
5-b) 0x0E / 0x02	(Not Active)	(Not Active)	(Active)	Change communication protocol
5-c) 0x0E / 0x03	(Not Active)	(Not Active)	(Active)	Change baud rate
6- Factory Reset Commands				
0x0F - Factory reset operations				
6-a) 0x0F / 0x01	(Not Active)	(Active)	(Active)	Restore all settings to factory defaults
6-b) 0x0F / 0x02	(Not Active)	(Not Active)	(Active)	Reading ID of format RFID card
6-c) 0x0F / 0x03	(Not Active)	(Not Active)	(Active)	Format flag set/reset
6-d) 0x0F / 0x04	(Not Active)	(Not Active)	(Active)	Manually write Format ID to Flash memory

Table 32 Descriptions of Command Categories and Sub-Categories According to Operating Modes

7.2 LED and Buzzer Control Commands

LED and buzzer control commands are used to manage the system's visual and auditory alert mechanisms. These commands allow functions such as turning specific LEDs on or off or making them blink, while for the buzzer, they enable turning it on, off, or generating sound for a specified duration. The user can select the appropriate command according to the system's needs, ensuring full control over the LEDs and buzzer.

7.2.1 Device LED Control Commands

To fully control the LEDs on the device, it is necessary to examine the **LED number** and **LED control commands**. There are **two different LEDs** on the device. The colors of these LEDs vary depending on the **device version information**.

LED color information based on hardware versions:

- **For devices with hardware major version 1.0 and minor version 1.1 or higher: Red-Blue LEDs**
- **For lower versions: Red-Yellow LEDs**

In this command structure, **data (DATA) is not required**. **COMMAND_TYPE and SUB_COMMAND_TYPE fields** are used to control the device LEDs.

7.2.1.1 Device LED On/Off Commands

In this command structure, **LEDs are turned on or off according to their colors** on the device.

- **The 3rd byte of the sent message (Hex format) provides the LED ON/OFF command information.**
- **The 4th byte provides the LED number information.**
- **The 5th byte provides the ON/OFF information.**

Byte Position	Description
3rd Byte	Indicates the LED ON/OFF command information.
4th Byte	Indicates the LED number (can be 0x01 or 0x03).
5th Byte	Provides the ON/OFF information (0x00 = OFF, 0x01 = ON).

- **The 4th byte value** can be 0x01 or 0x03. If any other value is sent, the device returns the **ERR_LED_NUMBER_INVALID** error.
  (For this error, please refer to **“**6.4.3 LED and Indicator-Related Errors”.)
- **Hex value 0x01** represents the **Red LED**.
- **Hex value 0x03** represents the **Blue or Yellow LED**, depending on the **device version information**.
- **The 5th byte value** provides the **ON/OFF information**, which can be 0x00 or 0x01. If an invalid value is sent, the **ERR_INVALID_LED_STATE** error is returned.
  (For this error, please refer to **“**6.4.3 LED and Indicator-Related Errors”.)

Below are the **Device LED Control Commands**:

7.2.1.1.1 Blue or Yellow LED Control Commands

Sent Command Format

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)
0xAA	0x01	0x01	0x03 0x01	NONE	0x79 0x30

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.
DEVICE_ADDRESS 0x01
Contains the device address.
COMMAND_TYPE 0x01
The key code that defines the main type of command; it is the LED On/Off command.
SUB_TYPE 0x03
Represents the LED number (blue LED).
0x01
Indicates the LED On command. If the LED needs to be turned off, 0x00 should be sent as the 5th Byte hex value.
(In this case, remember that the CRC value will change.)
CRC (Optional)
The CRC returned in hexadecimal is 0x79 0x30.

7.2.1.1.2 Red LED Control Commands

Sent Message

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)
0xAA	0x01	0x01	0x01 0x00	NONE	0x0F 0x73

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x01
The key code that defines the main type of command; it is the LED On/Off command.

SUB_TYPE 0x01
Represents the LED number (red LED).

0x00
Indicates the LED Off command. If the LED needs to be turned on, 0x01 should be sent as the 5th Byte hex value.
(In this case, remember that the CRC value will change.)

CRC (Optional)
The CRC returned in hexadecimal is 0x0F 0x73.

,

Received Message

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent LED command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.

For a better understanding of general system status and errors in response messages, please refer to the **"**Status Messages (Report_status)**"** section.

7.2.2 Device LED Toggle Commands

This command structure is used to toggle the LEDs on the device according to their colors. **It performs a single LED toggle operation**.

The **3rd Byte (Hex format) of the sent message** indicates that the **toggle operation will be performed**.

The value for this operation is 0x02.

The **4th Byte (Hex format) of the sent message** provides the **LED number information**.

The **5th Byte (Hex format) contains the LED toggle command**, which is **always 0x00**.

The **4th Byte can be 0x01 or 0x03**. If any other value is sent, the device returns an **ERR_LED_NUMBER_INVALID** error.

(For this error, please refer to **"**6.4.3 LED and Indicator-Related Errors**"**.)

Hex value 0x01 represents the **Red LED**.

Hex value 0x03 represents the **Blue or Yellow LED**, depending on the **device version information**.

7.2.2.1 Blue or Yellow LED Toggle Commands

Sent Message

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)
0xAA	0x01	0x02	0x03 0x00	NONE	0x30 0x41

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x02
The key code that defines the main type of command; it is the **LED toggle command**.

SUB_TYPE 0x03
Represents the **LED number (Blue LED)**.

0x00
Indicates the **LED toggle command**.

Received Message,

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent LED command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.

For a better understanding of general system status and errors in response messages, please refer to the "Status Messages (Report_status)" section.

7.2.2.2 Red LED Toggle Commands

Gönderilen Mesaj

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)
0xAA	0x01	0x02	0x01 0x00	NONE	0x56 0x23

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x02
The key code that defines the main type of command; it is the **LED toggle command**.

SUB_TYPE 0x01 0x00
Represents the **LED number (Red LED)**.
Indicates the **LED toggle command**.

Received Message,

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent LED command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.

For a better understanding of general system status and errors in response messages, please refer to the "Status Messages (Report_status)" section.

7.2.3 Device LED Dual Toggle Commands

In this command structure, the LEDs on the device toggle sequentially.
Unlike the previous toggle operation, **this command toggles two LEDs simultaneously**.

The **3rd Byte (Hex format) of the sent message** indicates that the dual LED toggle operation will be performed.

The value for the toggle operation is 0x03.

The **4th Byte (Hex format) of the sent message** specifies the dual toggle operation:

0x01 starts the dual toggle operation.

0x00 stops the dual toggle operation.

7.2.3.1 Starting Device LED Dual Toggle

Sent Message

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)
0xAA	0x01	0x03	0x01 0x00	NONE	0x61 0x13

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x03
The key code that defines the main type of command; it is the **dual LED toggle command**.

SUB_TYPE 0x01 0x00
Indicates the **toggle start command**.
The next **two bytes** represent the CRC value.

Received Message

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent LED command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.

For a better understanding of general system status and errors in response messages, please refer to the "Status Messages (Report_status)" section.

7.2.3.2 Stopping Device LED Dual Toggle

Sent Message

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)
0xAA	0x01	0x03	0x00 0x00	NONE	0x52 0x22

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x03
The key code that defines the main type of command; it is the **dual LED toggle command**.

SUB_TYPE 0x00 0x00
Indicates the **toggle stop command**.
The next **two bytes** represent the CRC value.

Received Message

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent LED command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.
For a better understanding of general system status and errors in response messages, please refer to the "Status Messages (Report_status)" section.

7.2.4 Device All LEDs Off Commands

This command structure is used to turn off all LEDs on the device.

The **3rd Byte (Hex format) of the sent message** indicates that the toggle operation will be performed.

The value for this operation is 0x04.

The **4th Byte and 5th Byte (Hex format) of the sent message** are both set to 0x00.

Sent Message

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)
0xAA	0x01	0x04	0x00 0x00	NONE	0XD7 0xB2

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x04
The key code that defines the main type of command; it is the **command for turning off all LEDs**.

Received Message,

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent LED off command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.
For a better understanding of general system status and errors in response messages, please refer to the **"**Status Messages (Report_status)**"** section.

7.2.5 Device Buzzer On/Off Commands

This command structure is used to turn the buzzer on and off on the device.

The **3rd Byte (Hex format) of the sent message** provides the **buzzer on/off command information**.

The value for the **buzzer command is 0x05**.

The **4th Byte (Hex format)** provides the **buzzer status information**:

0x01 → Buzzer **ON**

0x00 → Buzzer **OFF**

7.2.5.1 Buzzer On/Off Command

Sent Message

Command Sent						
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)	State
0xAA	0x01	0x05	0x01 0x00	NONE	0xD3 0xB3	Buzzer OPEN
0xAA	0x01	0x05	0x00 0x00	NONE	0xE0 0x82	Buzzer OFF

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x05
The key code that defines the main type of command; it is the **Buzzer On/Off command**.

SUB_TYPE 0x01 0x01
Represents the **Buzzer ON Command**.

SUB_TYPE 0x00 0x00
Represents the **Buzzer OFF Command**.

CRC (Optional)

The CRC value for **Buzzer ON** is 0xD3 0xB3.

The CRC value for **Buzzer OFF** is 0xE0 0x82.

Received Message

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent buzzer command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.

For a better understanding of general system status and errors in response messages, please refer to the **"**Status Messages (Report_status)**"** section.

7.2.6 Timed Buzzer On/Off Commands

This command structure is used to turn the buzzer on and off for a specified duration.

The **buzzer can be programmed in 100 ms intervals**.

The **maximum buzzer duration is 25000 ms**.

Unlike other commands, the device returns **two "SUCCESS" messages** after executing this command:

1. The first "SUCCESS" message is sent when the buzzer is **activated**.
2. The buzzer remains active for the specified duration (in milliseconds).
3. The second "SUCCESS" message is sent when the buzzer is **deactivated**.

The **3rd Byte (Hex format) of the sent message** provides the **buzzer on/off command information**.

The value for the buzzer command is **0x06**.

The **4th Byte (Hex format)** is always **0x00**.

The **5th Byte (Hex format) provides the buzzer active duration** in hexadecimal.

This **hexadecimal value is multiplied by 100 ms** to determine the total buzzer activation duration.

Once the calculated duration elapses, the buzzer automatically turns off.

7.2.6.1 Timed Buzzer On/Off Command

Sent Message

Command Sent						
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (opsiyonel)	State
0xAA	0x01	0x06	0x00 0x05	NONE	0xE9 0x77	Buzzer 500ms
0xAA	0x01	0x06	0x00 0x0A	NONE	0x18 0x98	Buzzer 1000ms
0xAA	0x01	0x06	0x00 0x14	NONE	0xEB 0x67	Buzzer 2000ms
0xAA	0x01	0x06	0x00 0xFA	NONE	0xA7 0x22	Buzzer 25000ms

HEADER 0xAA
Indicates the beginning of the command. It takes a fixed value and is found at the start of all messages.

DEVICE_ADDRESS 0x01
Contains the device address.

COMMAND_TYPE 0x06
The key code that defines the main type of command; it is the **Buzzer On/Off command**.

SUB_TYPE 0x00 0xXX
This is the **Timed Buzzer On Command**.
The 0xXX hexadecimal value is multiplied by **100 ms** to determine the buzzer activation duration.

Hex Value	Decimal Value	Active Buzzer Time
0x05	5 X 100ms	500 ms
0x0A	10 X 100ms	1000ms
0x14	20 X 100ms	2000 ms
0xFF	250 X 100ms	25000 ms

Table 33 Timed Buzzer On/Off Command ExampleReceived Message

When a valid timed buzzer command is sent, the device **sends the response message twice**:

First response → Sent when the buzzer is activated.

Second response → Sent when the buzzer is deactivated.

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

The response message indicates a **successful operation** based on the sent buzzer command.
This message information can be understood through **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** values.

For a better understanding of general system status and errors in response messages, please refer to the "Status Messages (Report_status)" section.

7.2.7 All LED and Programmable Buzzer Commands

These commands are provided for devices with **device address 0x01** and **CRC calculation mode CRC-16 CCITT**.

Command Name	Data Packet	Description
LED1ON	0xAA 0x01 0x01 0x01 0x01 0x1F 0x52	Turns **LED1** ON.
LED1OFF	0xAA 0x01 0x01 0x01 0x00 0x0F 0x73	Turns **LED1** OFF.
LED2ON	0xAA 0x01 0x01 0x03 0x01 0x79 0x30	Turns **LED2** ON.
LED2OFF	0xAA 0x01 0x01 0x03 0x00 0x69 0x11	Turns **LED2** OFF.
L.1.TOG	0xAA 0x01 0x02 0x01 0x00 0x56 0x23	Toggles **LED1** (blinking mode).
L.2.TOG	0xAA 0x01 0x02 0x03 0x00 0x30 0x41	Toggles **LED2** (blinking mode).
2_LED_TOGGLE_ON	0xAA 0x01 0x03 0x01 0x00 0x61 0x13	Starts **police light effect** (LEDs blink alternately).
2_LED_TOGGLE_OFF	0xAA 0x01 0x03 0x00 0x00 0x52 0x22	Stops the **police light effect**.
LED_ALL_OFF	0xAA 0x01 0x04 0x00 0x00 0xD7 0xB2	Turns **off all LEDs** in the system.
BZR_ON	0xAA 0x01 0x05 0x01 0x00 0xD3 0xB3	Turns the **buzzer ON** (starts sound output).
BZR_OFF	0xAA 0x01 0x05 0x00 0x00 0xE0 0x82	Turns the **buzzer OFF**.
B_1000	0xAA 0x01 0x06 0x00 0x0A 0x18 0x98	Activates the buzzer for **1000 ms (1 second)**.
B_500	0xAA 0x01 0x06 0x00 0x05 0xE9 0x77	Activates the buzzer for **500 ms (half a second)**.
B_2000	0xAA 0x01 0x06 0x00 0x14 0xEB 0x67	Activates the buzzer for **2000 ms (2 seconds)**.
B_25000	0xAA 0x01 0x06 0x00 0xFF 0xA7 0x22	Activates the buzzer for **25000 ms (25 seconds)**

Table 34 LED and Buzzer Control Commands Table

This table provides an overview of **all LED and buzzer control commands** along with their corresponding data packets and functionalities.

7.3 RFID Card Operation Commands

RFID card operation commands include the essential command set used for card recognition, reading, and writing operations. These commands ensure secure and efficient communication with RFID systems. The commands listed below cover various functions, including reading the card's **UID (Unique Identifier) number**, **SAK (Select Acknowledge)**, and **ATQA (Answer To Request) values**, as well as reading and writing data from specific blocks. Additionally, security-focused operations such as displaying system-defined keys are included within this command set.

RFID card operation commands play a crucial role in authentication processes while ensuring secure data management. Each command's function is explained in detail below.

7.3.1 RFID Card Recognition and Read/Write Commands

RFID card reading operations consist of essential commands that allow the device to recognize RFID cards and retrieve necessary data. These commands are designed to obtain information such as the card's **UID (Unique Identifier)**, **SAK (Select Acknowledge)**, and **ATQA (Answer To Request) values**. Additionally, data reading and writing from specific blocks are supported.

Commands follow a **dynamic-length structure** consisting of **HEADER, DEVICE_ADDRESS, COMMAND_TYPE, SUB_COMMAND_TYPE, DATA (only for read/write operations), and CRC (optional)** components.

The DATA field is only used in RFID card block read and write commands.

The CRC field, used for ensuring data integrity, is optionally appended at the end of the message.

Section	Description
HEADER	Determines the start of the command. It is a fixed value and is always at the beginning of the message.
DEVICE_ADDRESS	Specifies the target device to which the command is sent. It contains the device address (0-255).
COMMAND_TYPE	Specifies the main command type used for RFID card operations.
SUB_COMMAND_TYPE	In RFID commands, it indicates different sub-commands such as UID, SAK, or ATQA reading.
DATA (If Applicable)	Used for reading and writing data from specific blocks. The DATA field is usually left empty for reading operations but contains the necessary data for writing operations.
CRC (Optional)	An error-checking mechanism used to ensure data integrity. If enabled, it is appended at the end of the message.

This structure allows **RFIDAX** devices to perform **RFID card reading and writing operations** reliably and flexibly.

Response Messages

Responses are received **in data packet format.**

Please carefully review the Data Packets section (6.7.2 Data Packets (send_data_packet)).

HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xAA	0x00-0xFF 1 Byte	Değişken 1 Byte	Variable Length	Variable (1 Byte)	Variable (1 Byte)

Note:

Before reviewing this section, referring to the **Data Packet Structure** (6.7.2 Data Packets (send_data_packet)) and **Message Type Information** (6.7.2.1 Message Type Information) will help you better understand the command structure.

7.3.2 RFID Card Recognition Commands

Certain subcommands are used to recognize RFID cards and read their contained information. These commands allow the reading of the **unique identifier (UID)**, security-related values such as **SAK (Select Acknowledge)** and **ATQA (Answer To Request)**, and general card information.
RFID card operations begin with **COMMAND_TYPE = 0x07**. The **SUB_COMMAND_TYPE** field determines which information will be read. During reading operations, the **UID, SAK, and ATQA** values can be retrieved separately, or all card information can be fetched in a single command.
In this command structure, the **DATA field is only used for specific block read and write operations**. For **UID, SAK, and ATQA** reading commands, the **DATA field remains empty**. The **CRC field is optional and may be added at the end of the message to ensure data integrity**.

7.3.2.1 RFID Card UID Reading Command

This command is used to read the **unique identification number (UID)** of an RFID card.

Sent Message Format

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (Opsiyonel)
0xAA	0x01	0x07	0x01	0xFF	0xA3 0x23

Field Descriptions

- **HEADER (0xAA):** Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the target device receiving the command.
- **COMMAND_TYPE (0x07):** Specifies that the operation is an RFID card process.
- **SUB_COMMAND_TYPE (0x01):** Specifies the operation of reading the **UID**.
- **DATA (0xFF):** This field is not used in this command but is sent as a placeholder value.
- **CRC (0xA3 0x23) (Optional):** Used for error checking and ensuring data integrity.

Received Message Format,

HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Kart ID)	CRC_HIGH (Opsiyonel)	CRC_LOW (Opsiyonel)
0xAA	0x01	0x07	0x88 0x04 0xDE 0x18	0xC4	0xD1

Response Field Descriptions

- **HEADER (0xAA):** Identifies the start of the response. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the responding device.
- **MESSAGE_TYPE (0x07):** Indicates that the response is an **RFID card operation response**.
- **DATA (Card UID):**
  - 0x88 → Indicates successful operation
  - 0x04 → Length of the UID (2 bytes)
  - 0xDE 0x18 → The UID value read from the RFID card
- **CRC_HIGH (0xC4) & CRC_LOW (0xD1) (Optional):** Used for error checking and ensuring data integrity.

This response confirms that the **RFID card UID has been successfully read**.
To better understand the MESSAGE_TYPE field, please refer to section "6.7.2.1.1 RFID Card Reading Operation Messages and Message Type Information."

7.3.2.2 RFID Card SAK Value Reading Command

This command is used to read the **SAK (Select Acknowledge) value** of an RFID card. The **SAK value is an authentication parameter that defines the card type and supported protocols**.

Sent Message Format

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (Opsiyonel)
0xAA	0x01	0x07	0x02	0xFF	0xF6 0x70

Field Descriptions

- **HEADER (0xAA):** Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the target device receiving the command.
- **COMMAND_TYPE (0x07):** Specifies that the operation is an **RFID card process**.
- **SUB_COMMAND_TYPE (0x02):** Specifies the operation of **reading the SAK value**.
- **DATA (0xFF):** This field is **not used in this command**, but it is sent as a placeholder value.
- **CRC (0xF6 0x70) (Optional):** Used for **error checking and ensuring data integrity**.

Received Message Format

Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Kart SAK Değeri)	CRC_HIGH (Opsiyonel)	CRC_LOW (Opsiyonel)
0xAA	0x01	0x09	0x08	0x0A	0xBD

Response Field Descriptions

- **HEADER (0xAA):** Identifies the start of the response. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the responding device.
- **MESSAGE_TYPE (0x09):** Indicates that the response is an **RFID card operation response**.
- **DATA (Card SAK Value):**
  - 0x08 → SAK value of the RFID card
- **CRC_HIGH (0x0A) & CRC_LOW (0xBD) (Optional):** Used for **error checking and ensuring data integrity**.

This response confirms that the **RFID card SAK value has been successfully read**.

To better understand the MESSAGE_TYPE field, please refer to section "6.7.2.1.1 RFID Card Reading Operation Messages **and** Message Type **Information."**

7.3.2.3 RFID Card ATQA Value Reading Command

This command is used to read the **ATQA (Answer To Request) value** of an RFID card.
The **ATQA value is a key identifier that determines the type of the card and its communication characteristics**.

Sent Message Format

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (Opsiyonel)
0xAA	0x01	0x07	0x03	0xFF	0xC5 0x41

Field Descriptions

- **HEADER (0xAA):** Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the target device receiving the command.
- **COMMAND_TYPE (0x07):** Specifies that the operation is an **RFID card process**.
- **SUB_COMMAND_TYPE (0x03):** Specifies the operation of **reading the ATQA value**.
- **DATA (0xFF):** This field is **not used in this command**, but it is sent as a placeholder value.
- **CRC (0xC5 0x41) (Optional):** Used for **error checking and ensuring data integrity**.

Received Message Format

Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Kart ATQA Değeri)	CRC_HIGH (Opsiyonel)	CRC_LOW (Opsiyonel)
0xAA	0x01	0x08	0x00 0x04	0xE2	0x57

Response Field Descriptions

- **HEADER (0xAA):** Identifies the start of the response. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the responding device.
- **MESSAGE_TYPE (0x08):** Indicates that the response is an **RFID card operation response**.
- **DATA (Card ATQA Value):**
  - 0x00 0x04 → ATQA value of the RFID card
- **CRC_HIGH (0xE2) & CRC_LOW (0x57) (Optional):** Used for **error checking and ensuring data integrity**.

This response confirms that the **RFID card ATQA value has been successfully read**.
To better understand the MESSAGE_TYPE field, please refer to section "6.7.2.1.1 RFID Card Reading Operation Messages** and **Message Type** Information."**

7.3.2.4 Read Card Information (UID + SAK + ATQA) Command

This command is used to **read all identification details of an RFID card in a single operation**.
It returns a comprehensive response that includes the **UID (Unique Identifier), SAK (Select Acknowledge), and ATQA (Answer To Request) values**.

Sent Message Format

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA	CRC (Opsiyonel)
0xAA	0x01	0x07	0x04	0xFF	0x5C 0xD6

Field Descriptions

- **HEADER (0xAA):** Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the target device receiving the command.
- **COMMAND_TYPE (0x07):** Specifies that the operation is an **RFID card process**.
- **SUB_COMMAND_TYPE (0x04):** Specifies the operation of **reading complete card information (UID + SAK + ATQA)**.
- **DATA (0xFF):** This field is **not used in this command**, but it is sent as a placeholder value.
- **CRC (0x5C 0xD6) (Optional):** Used for **error checking and ensuring data integrity**.

Received Message Format

Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Kart UID + SAK + ATQA)	CRC_HIGH (Opsiyonel)	CRC_LOW (Opsiyonel)
0xAA	0x01	0x0A	0x66 0xA7 0x7B 0xDA 0x08 0x00 0x04	0x31	0x3C

Response Field Descriptions

- **HEADER (0xAA):** Identifies the start of the response. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the responding device.
- **MESSAGE_TYPE (0x0A):** Indicates that the response is an **RFID card operation response**.
- **DATA (Card UID + SAK + ATQA):**
  - **0x66 0xA7 0x7B 0xDA → Card UID (Unique Identifier Number)**
  - **0x08 → Card SAK Value**
  - **0x00 0x04 → Card ATQA Value**
- **CRC_HIGH (0x31) & CRC_LOW (0x3C) (Optional):** Used for **error checking and ensuring data integrity**.

This response confirms that the **RFID card UID, SAK, and ATQA values have been successfully read**.
To better understand the MESSAGE_TYPE field, please refer to section "6.7.2.1.1 RFID Card Reading Operation Messages and Message Type Information."

7.3.2.5 RFID Card Key Display Command

This command is used to **display the security keys (KeyA, KeyB, Optional KeyA, Optional KeyB) of an RFID card**.
In RFID systems, access to card blocks is controlled using specific keys. **This command allows reading and verifying the stored key information**.
If the **SUB_COMMAND_TYPE** value is:

- **0x05 0x01** → Displays **KeyA**
- **0x05 0x02** → Displays **KeyB**
- **0x05 0x03** → Displays **Optional KeyA**
- **0x05 0x04** → Displays **Optional KeyB**
- **0x05 0xFF** → Displays **all keys**
- **KeyA and KeyB** are **fixed keys** in RFID systems and **cannot be changed**.
- If a **custom key** is required, **Optional KeyA and Optional KeyB** can be **stored, modified, and read by the user**.

Sent Message Format

Command Sent						
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	DATA (Not Used)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x07	0x05 XX	Not Used	XX	XX

Field Descriptions

- **HEADER (0xAA):** Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the target device receiving the command.
- **COMMAND_TYPE (0x07):** Defines RFID card operations.
- **SUB_COMMAND_TYPE:** Determines which key is read:
  - **0x05 0x01 → KeyA**
  - **0x05 0x02 → KeyB**
  - **0x05 0x03 → Optional KeyA**
  - **0x05 0x04 → Optional KeyB**
  - **0x05 0xFF → Displays all keys**
- **DATA: Not used in this command.**
- **CRC (XX XX) (Optional):** Used to ensure data integrity

7.3.2.5.1 Fixed Keys (KeyA and KeyB)

In RFID systems, **KeyA and KeyB** are used to control access to card blocks.
These keys are **fixed, cannot be changed**, and are **predefined in the system as default authentication keys**.

Key Type	Default Value
KeyA	0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
KeyB	0x00 0x00 0x00 0x00 0x00 0x00

KeyA and KeyB are fixed keys defined by the manufacturer and cannot be modified.
They are generally used as default access keys.

KeyA Reading

To read **KeyA**, please refer to the following command structure.

Sent Message (KeyA Reading - Fixed Key)

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	CRC_HIGH	CRC_LOW
0xAA	0x01	0x07	0x05 0x01	0x61	0x36

Received Message (KeyA Response - Fixed Key)

Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (KeyA - Fixed Key)	CRC_HIGH	CRC_LOW
0xAA	0x01	0x0B	0xFF 0xFF 0xFF 0xFF 0xFF 0xFF	0x74	0x87

KeyB Reading

To read **KeyB,** please refer to the following command structure.

Sent Message (KeyB Reading - Fixed Key)Command Sent

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	CRC_HIGH	CRC_LOW
0xAA	0x01	0x07	0x05 0x02	0x51	0x55

Received Message (KeyB Response - Fixed Key)

Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (KeyB - Fixed Key)	CRC_HIGH	CRC_LOW
0xAA	0x01	0x0C	0x00 0x00 0x00 0x00 0x00 0x00	0xFA	0x1C

7.3.2.5.2 User-Defined Keys (Optional KeyA and Optional KeyB)

For special security requirements, **user-defined keys** are available.
These keys can be **stored, modified, and read by the user**.

Key Type	Example User-Saved Value
Optional KeyA	0x1A 0x2A 0x3A 0x4A 0x5A 0x6A
Optional KeyB	0xA1 0xB2 0x99 0xD1 0xE1 0xF1

Optional KeyA and Optional KeyB can be stored and modified by the user.
If a special KeyA or KeyB is needed, optional keys should be used.
The modification process for these keys is explained in the "Memory Management Commands" section. Please refer to 7.5.2 OptionalKeyA and OptionalKeyB Operations)

Optional KeyA Reading

To read **Optional KeyA**, please refer to the following command structure

Sent Message (Optional KeyA Reading - User-Saved Key)

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	CRC_HIGH	CRC_LOW
0xAA	0x01	0x07	0x05 0x03	0x41	0x74

Received Message (Optional KeyA Response - User-Saved Key)

Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Optional KeyA - User-Saved Key)	CRC_HIGH	CRC_LOW
0xAA	0x01	0x0E	0x1A 0x2A 0x3A 0x4A 0x5A 0x6A	0xFA	0xD5

Optional KeyB Reading

To read **Optional KeyB**, please refer to the following command structure.

Sent Message (Optional KeyB Reading - User-Saved Key)

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	CRC_HIGH	CRC_LOW
0xAA	0x01	0x07	0x05 0x04	0x31	0x93

Received Message (Optional KeyB Response - User-Saved Key)

Response					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Optional KeyB - User-Saved Key)	CRC_HIGH	CRC_LOW
0xAA	0x01	0x0F	0xA1 0xB2 0x99 0xD1 0xE1 0xF1	0x57	0x5A

7.3.2.5.3 Reading All Keys (KeyA, KeyB, Optional KeyA, and Optional KeyB)

To read all keys (**KeyA, KeyB, Optional KeyA, and Optional KeyB**), please refer to the following command structure.

Sent Message (Reading All Keys - FF Usage)

Command Sent					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	CRC_HIGH	CRC_LOW
0xAA	0x01	0x07	0x05 0xFF	0x6F	0xE7

Received Message (Response for Retrieving All Keys - FF Usage)

When this command is sent, the device returns **four separate messages** in the following order:

Response Type	Message
KeyA Response	0xAA 0x01 0x0B 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x74 0x87
KeyB Response	0xAA 0x01 0x0C 0x00 0x00 0x00 0x00 0x00 0x00 0xFA 0x1C
Optional KeyA Response	0xAA 0x01 0x0E 0x1A 0x2A 0x3A 0x4A 0x5A 0x6A 0xFA 0xD5
Optional KeyB Response	0xAA 0x01 0x0F 0xA1 0xB2 0x99 0xD1 0xE1 0xF1 0x57 0x5A

The modification process for these keys is explained in the "Memory Management Commands" section. Please refer to (7.5.2 OptionalKeyA and OptionalKeyB Operations)

7.3.3 Writing and Reading Data from Blocks in RFID Cards

RFID cards contain **data storage areas consisting of specific memory blocks**.
These blocks can be **written to, read from, and protected with specific access levels**.
This section details the **processes of writing data to blocks and reading data from blocks** in RFID cards.
RFID block reading and writing operations **require authentication** and access is provided using **KeyA, KeyB, Optional KeyA, or Optional KeyB**.
Each block has a **specific security structure** and requires **authentication**.
Access and write permissions for blocks depend on the **security configuration of the card**.

7.3.3.1 RFID Card Block Write Command

This command is used to write data to specific blocks on an RFID card.
During the writing process, **authentication with KeyA, KeyB, Optional KeyA, or Optional KeyB is mandatory**.
The **start and end block addresses** must be specified, and the card's **security structure must be checked** for compliance.
If an **invalid block is selected** or if the **data exceeds the permitted limits**, the operation is rejected, and an **error message is returned**.
Command Sent (RFID Card Block Write Command)

Command Sent											
HEADER	DEVICE_ ADDRESS	COMMAND_ TYPE	SUB_ COMMAND_TYPE	KEY _TYPE	AUTH _TYPE	START _BLOCK	END _BLOCK	FORMAT_TYPE	DATA (Variable Length)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x09	0xXX	0x01 0x02 0x03 or 0x04	0x01 or 0x02	0xXX	0xXX	0x00	User Data	XX	XX

Field Descriptions

- **HEADER (0xAA):** Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** Specifies the address of the device receiving the command.
- **COMMAND_TYPE (0x09):** Defines RFID card operations. This command refers to the **block write operation**.
- **SUB_COMMAND_TYPE (0xXX):** Not used in this command.
- **KEY_TYPE (0x01, 0x02, 0x03, 0x04):** Specifies the type of key used.
  - 0x01 → KeyA
  - 0x02 → KeyB
  - 0x03 → Optional KeyA
  - 0x04 → Optional KeyB
- **AUTH_TYPE (0x01 or 0x02):** Specifies the authentication method.
  - 0x01 → PICC_AUTHENT1A
  - 0x02 → PICC_AUTHENT1B
- **START_BLOCK (0xXX):** Specifies the **first block** where data will be written.
- **END_BLOCK (0xXX):** Specifies the **last block** where data will be written.
  - If **writing to a single block**, the **START_BLOCK and END_BLOCK values must be the same**.
- **FORMAT_TYPE (0x00 or 0x01):** Specifies the format of the data.
  - 0x00 → HEX format
  - 0x01 → ASCII format (This format is not used)
- **DATA (Variable Length):** The data to be written by the user.
- **CRC_HIGH (XX) & CRC_LOW (XX) (Optional):** Error control mechanism used to ensure data integrity.

Response (RFID Card Block Write Response)

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (opsiyonel)	CRC_LOW (opsiyonel)
0xBB	0x01	0x00	0x00	0x5C	0x3E

Field Descriptions

HEADER (0xBB): Identifies the start of the message. It is a fixed value.
DEVICE_ADDRESS (0x01): Indicates the device that sends the response.
STATUS:

0x00 → Successfully written.
Error Code → Writing process failed.

CRC_HIGH (XX) & CRC_LOW (XX) (Optional): Error control mechanism used to ensure data integrity.

The response message for the write operation confirms successful execution.
This information can be understood from the **"STATUS_CODE_HIGH and STATUS_CODE_LOW"** fields.
For a better understanding of general system status and errors, please refer to Section 6.4.4 RFID Card Read/Write Errors.

Operation Workflow of This Command

1. The presence of the card is checked.
   If the card is **not detected**, the system returns ERR_CARD_NOT_FOUND.
2. Start and end block addresses are validated.
   If the **start block is greater than the end block**, ERR_INVALID_BLOCK_RANGE is returned.
3. Authentication is performed.
   If authentication using **KeyA, KeyB, Optional KeyA, or Optional KeyB** fails, ERR_AUTHENTICATION_FAILED is returned.
4. Data format validation is done.
   If the **format is HEX**, the data is directly allowed for writing.
5. Data length is checked.
   If the **data is too long for the selected blocks**, ERR_DATA_LENGTH_EXCEEDED is returned.
6. Writing process starts.
   Data is written in **16-byte blocks**.
   Block 0 and control blocks of each sector (e.g., **3, 7, 11**) are skipped.
7. Once the operation is complete, the card is halted, and the module is restarted.

Possible Error Codes

Error Code	Description
ERR_CARD_NOT_FOUND	Card was not detected, operation cannot proceed.
ERR_INVALID_BLOCK_RANGE	Invalid block range selected.
ERR_INVALID_DATA	Data to be written is incorrect or incomplete.
ERR_DATA_LENGTH_EXCEEDED	Data is too long, does not fit in the selected blocks.
ERR_AUTHENTICATION_FAILED	Authentication failed, authorization could not be completed.

Summary

This command is used to write data to specific blocks on an RFID card.
Authentication with KeyA, KeyB, Optional KeyA, or Optional KeyB is required.
Data must be in HEX format.
Invalid block ranges, authentication failures, or data size issues can result in failed operations.
This command is crucial for advanced access control and secure data storage on RFID cards.

7.3.3.1.1 Example of Writing Data to RFID Card Blocks

This example demonstrates the **process of writing data to a specific block** on an RFID card.
The write operation is performed by specifying **authentication (KeyA or KeyB or optional_ KeyA or optional_ KeyB), start and end block addresses, and data format**.

Sent Message Format

Command Sent (Command to Write Data to RFID Card Blocks)										
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	KEY _ TYPE	AUTH _ TYPE	START _ BLOCK	END _ BLOCK	FORMAT _TYPE	DATA (Variable Length)	CRC _HIGH (Optional)	CRC _LOW (Optional)
0xAA	0x01	0x09	0x01	0x02	0x21	0x21	0x00	0x52 0x46 0x49 0x44 0x41 0x58 0x20 0x44 0x65 0x76 0x69 0x63 0x65 0x73	0x24	0xBB

DATA (Variable Length) Açıklamaları

HEX Format	ASCII Equivalent
0x52	R
0x46	F
0x49	I
0x44	D
0x41	A
0x58	X
0x20	(Space)
0x44	D
0x65	e
0x76	v
0x69	i
0x63	c
0x65	e
0x73	s

HEX Format: 0x52 0x46 0x49 0x44 0x41 0x58 0x20 0x44 0x65 0x76 0x69 0x63 0x65 0x73
ASCII Equivalent: " RFIDAX Devices "

Field Descriptions

HEADER (0xAA): Identifies the start of the command. It is a fixed value.
DEVICE_ADDRESS (0x01): Specifies the address of the device receiving the command.
COMMAND_TYPE (0x09): Defines RFID card operations (**RFID block write command**).
KEY_TYPE (0x01): Specifies that authentication will be performed using **KeyA**.
AUTH_TYPE (0x02): Specifies the **PICC_AUTHENT1B** authentication method.
START_BLOCK (0x21): Specifies the **first block address** to be written.
END_BLOCK (0x21): Specifies the **last block address** to be written (**only a single block is being written**).
FORMAT_TYPE (0x00): Data format is selected as **HEX**.
DATA: "RFIDAX Devices" (**ASCII characters represented in HEX format**).
CRC_HIGH (0x24) & CRC_LOW (0xBB) (Optional): Error control mechanism used to ensure data integrity.

Response Message Format

Response (RFID Card Block Write Response)					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xBB	0x01	0x00	0x00	0x5C	0x3E

Field Descriptions

HEADER (0xBB): Identifies the start of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01): Indicates the device that sends the response.
STATUS_CODE_HIGH (0x00): The first byte indicating the operation result.
STATUS_CODE_LOW (0x00): The second byte indicating the operation result.
0x00 0x00 → Successfully written.
CRC_HIGH (0x5C) & CRC_LOW (0x3E) (Optional): Error control mechanism used to verify data integrity.

Processes Executed in This Example

1. **The presence of the RFID card was verified.**
2. **Authentication was performed** using **KeyA** and **AUTH_TYPE = PICC_AUTHENT1B**.
3. **The text "RFIDAX Cihazlari" was written in HEX format to block 0x21.**
4. **The device successfully completed the write operation** and returned the **0x00 0x00 (success) response.**

Summary

In this example, **ASCII format data "** RFIDAX Devices **" was written to block 0x21 on the RFID card.**
The device successfully completed the write operation and returned a **success response (0x00 0x00).**
This message format serves as an essential reference for understanding **RFID block writing operations.**

7.3.3.1.2 Example of Writing Data to RFID Card Blocks – 2

This example demonstrates the **process of writing data to a specific block range** on an RFID card. The write operation is performed by specifying **authentication (KeyA or KeyB or optional_ KeyA or optional_ KeyB), start and end block addresses, and data format**.

Sent Message Format

Command Sent (RFID Kart Bloklarına Veri Yazma Komutu)
AA 01 09 01 02 04 06 00 4F 6E 75 72 20 4B 61 67 61 6E 20 45 73 6D 65 72 61 79 20 48 61 72 64 77 61 72 65 20 45 6E 67 69 6E 65 65 72 20 20 EC DB

HEADER	DEVICE _ADDRESS	COMMAND _TYPE	KEY _TYPE	AUTH _TYPE	START _BLOCK	END _BLOCK	FORMAT _TYPE	DATA (Variable Length)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x09	0x01	0x02	0x04	0x06	0x00	See Below	0xEC	0xDB

DATA (Variable Length) Explanation

HEX Format:
0x4F 0x6E 0x75 0x72 0x20 0x4B 0x61 0x67 0x61 0x6E 0x20 0x45 0x73 0x6D 0x65 0x72 0x61 0x79 0x20 0x48 0x61 0x72 0x64 0x77 0x61 0x72 0x65 0x20 0x45 0x6E 0x67 0x69 0x6E 0x65 0x65 0x72 0x20 0x20

ASCII Equivalent: "Onur Kagan Esmeray Hardware Engineer "

Field Descriptions

HEADER (0xAA): Identifies the start of the command. It is a fixed value.
DEVICE_ADDRESS (0x01): Specifies the address of the device receiving the command.
COMMAND_TYPE (0x09): Defines RFID card operations (**RFID block write command**).
KEY_TYPE (0x01): Specifies that authentication will be performed using **KeyA**.
AUTH_TYPE (0x02): Specifies the **PICC_AUTHENT1B** authentication method.
START_BLOCK (0x04): Specifies the **first block address** to be written.
END_BLOCK (0x06): Specifies the **last block address** to be written.
FORMAT_TYPE (0x00): Data format is selected as **HEX**.
DATA: "Onur Kagan Esmeray Hardware Engineer " (**ASCII characters represented in HEX format**).
CRC_HIGH (0xEC) & CRC_LOW (0xDB) (Optional): Error control mechanism used to ensure data

Response Message Format

Response (RFID Card Block Write Response)

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xBB	0x01	0x00	0x00	0x5C	0x54

Field Descriptions

HEADER (0xBB): Identifies the start of the response message. It is a fixed value.

DEVICE_ADDRESS (0x01): Indicates the device that sends the response.

STATUS_CODE_HIGH (0x00): The first byte indicating the operation result.

STATUS_CODE_LOW (0x00): The second byte indicating the operation result.

 0x00 0x00 → Successfully written.

CRC_HIGH (0x5C) & CRC_LOW (0x54) (Optional): Error control mechanism used to verify data integrity.

Processes Executed in This Example

- The presence of the RFID card was verified.
- Authentication was performed using KeyA and AUTH_TYPE = PICC_AUTHENT1B.
- The text "Onur Kagan Esmeray Hardware Engineer" was written in HEX format to blocks 0x04 - 0x06.
- The device successfully completed the write operation and returned the 0x00 0x00 (success) response.

Summary

- In this example, **ASCII format data "Onur Kagan Esmeray Hardware Engineer" was written to blocks 0x04 - 0x06 on the RFID card.**
- The device successfully completed the write operation and returned a **success response (0x00 0x00).**
- This message format serves as an essential reference for understanding **RFID block writing operations.**

7.3.3.1.3 Example of Writing Data to RFID Card Blocks – 3

This example demonstrates writing long data to multiple blocks of an RFID card. Authentication (**KeyA, KeyB, Optional_KeyA, Optional_KeyB**) is performed, start and end block addresses are defined, and long text data is written in **HEX format**.

Sent Message Format

Command Sent (Writing Long Data to RFID Card Blocks)

Command Sent (Writing Long Data to RFID Card Blocks)										
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	KEY _TYPE	AUTH _TYPE	START _BLOCK	END _BLOCK	FORMAT _TYPE	DATA (Variable Length)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x03	0x09	0x01	0x02	0x10	0x1A	0x00	See Below	0x45	0xF0

DATA (Variable Length) Description

HEX Format:
0x52 0x46 0x49 0x44 0x41 0x58 0x20 0x64 0x65 0x76 0x69 0x63 0x65 0x73 0x20 0x61 0x72 0x65 0x20 0x74 0x68 0x65 0x20 0x62 0x65 0x73 0x74 0x20 0x61 0x6E 0x64 0x20 0x6D 0x6F 0x73 0x74 0x20 0x74 0x65 0x63 0x68 0x6E 0x6F 0x6C 0x6F 0x67 0x69 0x63 0x61 0x6C 0x20 0x64 0x65 0x76 0x69 0x63 0x65 0x73 0x20 0x69 0x6E 0x20 0x74 0x68 0x65 0x20 0x69 0x6E 0x64 0x75 0x73 0x74 0x72 0x79 0x2E 0x20 0x54 0x68 0x65 0x72 0x65 0x20 0x61 0x72 0x65 0x20 0x6D 0x61 0x6E 0x79 0x20 0x6D 0x6F 0x64 0x65 0x6C 0x73 0x2E 0x20 0x46 0x6F 0x72 0x20 0x6F 0x72 0x64 0x65 0x72 0x69 0x6E 0x67 0x2C 0x20 0x76 0x69 0x73 0x69 0x74 0x20 0x77 0x77 0x77 0x2E 0x72 0x66 0x69 0x64 0x61 0x78 0x2E 0x63 0x6F 0x6D

ASCII Equivalent:
"RFIDAX devices are the best and most technological devices in the industry. There are many models. For ordering, visit www.rfidax.com"

Field Descriptions

HEADER (0xAA): Identifies the start of the command. It is a fixed value.
DEVICE_ADDRESS (0x03): Specifies the address of the device receiving the command.
COMMAND_TYPE (0x09): Defines RFID card operations (**RFID block write command**).
KEY_TYPE (0x01): Specifies that authentication will be performed using **KeyA**.
AUTH_TYPE (0x02): Specifies the **PICC_AUTHENT1B** authentication method.
START_BLOCK (0x10): Specifies the first block address to be written.
END_BLOCK (0x1A): Specifies the last block address to be written.
FORMAT_TYPE (0x00): Data format is selected as **HEX**.
DATA: "RFIDAX devices are the best and most technological devices in the industry. There are many models. For ordering, visit www.rfidax.com" (**ASCII characters represented in HEX format**).
CRC_HIGH (0x45) & CRC_LOW (0xF0) (Optional): Error control mechanism used to ensure data integrity.

Response Message Format

Response (RFID Card Block Write Response)

Response					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xBB	0x03	0x00	0x00	0x32	0x5E

Field Descriptions

HEADER (0xBB): Identifies the start of the response message. It is a fixed value.

DEVICE_ADDRESS (0x03): Indicates the device that sends the response.

STATUS_CODE_HIGH (0x00): The first byte indicating the operation result.

STATUS_CODE_LOW (0x00): The second byte indicating the operation result.

0x00 0x00 → Successfully written.

CRC_HIGH (0x32) & CRC_LOW (0x5E) (Optional): Error control mechanism used to verify data integrity.

Processes Executed in This Example

The presence of the RFID card was verified.

Authentication was performed using KeyA and AUTH_TYPE = PICC_AUTHENT1B.

The text **"RFIDAX devices are the best and most technological devices in the industry. There are many models. For ordering, visit www.rfidax.com"** was written in **HEX format** to blocks **0x10 - 0x1A**.

The device successfully completed the write operation and returned the 0x00 0x00 (success) response.

Summary

In this example, **ASCII format data** "RFIDAX devices are the best and most technological devices in the industry. There are many models. For ordering, visit www.rfidax.com" was written to blocks **0x10 - 0x1A** on the RFID card.

The device successfully completed the write operation and returned a success response **(0x00 0x00)**.

This message format serves as an **essential reference** for understanding **RFID block writing operations**.

7.3.3.2 RFID Card Block Read Command

It is the command used to read data written on certain blocks on RFID cards. The read operation **requires authentication (KeyA, KeyB, Optional KeyA, Optional KeyB) and data is pulled** from the block addresses specified by **the user and** sent back in response.
This command **can be used to read** a single block **or** a specific range of blocks.**The starting block and ending block** are determined and the data in the card is read.
If the block interval **is invalid**, i.e. the starting block is greater than the end block, an error message is returned.
After the read operation is complete, the system **returns** the read block data in 16-byte packets.

Sent Message Format
Command Sent (RFID Card Block Read Command)

Command Sent (Command to Read Data to RFID Card Blocks)								
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	KEY _TYPE	AUTH _TYPE	START _BLOCK	END _BLOCK	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x08	0x01 - 0x04	0x01 - 0x02	0xXX	0xXX	XX	XX

Field Descriptions

- **HEADER (0xAA):** Specifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** The address of the device from which the command was sent.
- **COMMAND_TYPE (0x08):** Determines the RFID block reading process.
- **KEY_TYPE (0x01 to 0x04):** Specifies the authentication key to use.
  - 0x01 → KeyA
  - 0x02 → KeyB
  - 0x03 → Optional KeyA
  - 0x04 → Optional KeyB
- **AUTH_TYPE (0x01 - 0x02):** Specifies the authentication method.
  - 0x01 → PICC_AUTHENT1A
  - 0x02 → PICC_AUTHENT1B
- **START_BLOCK (0xXX):** Specifies the first block address to read.
- **END_BLOCK (0xXX):** Specifies the last block address to read. **If a single block is to be read, the START_BLOCK and END_BLOCK must be the same.**
- **CRC_HIGH (XX) & CRC_LOW (XX) (Optional):** It is the error control mechanism used to ensure data integrity.

Response Message Format

Response (Data Response Read from RFID Card Blocks)

Response (Data Response Read from RFID Card Blocks)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Block Data - 16 Bytes)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x10	16 Byte Blok Verisi	0xXX	0xXX

Field Descriptions

HEADER (0xAA): Specifies the beginning of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01): Shows the address of the device that sent the response.
MESSAGE_TYPE (0x10): Indicates that there is an RFID block read response.
DATA (Block Data - 16 Bytes): Contains the RFID block data read. For each block, 16 bytes of data are returned.
CRC_HIGH (0xXX) & CRC_LOW (0xXX) (Optional): It is the error control mechanism used to ensure data integrity.

The principle of operation of this command

The presence of the RFID card is checked. If the card is not detected, a ERR_CARD_NOT_FOUND error is returned.
The start and end blocks are verified. If the starting block is greater than the ending block, a ERR_INVALID_BLOCK_RANGE error is returned.
Authentication is done.
If validation fails, a ERR_AUTHENTICATION_FAILED error is returned.
Verification is done with KeyA, KeyB, Optional KeyA or Optional KeyB.
The block read process is initiated.
If a single block is to be read, only START_BLOCK is read.
If a range of blocks is to be read, all blocks between START_BLOCK and END_BLOCK are read.
The read data is returned in response.
For each block, 16 bytes of data are returned.

Possible Error Codes	
Error Code	Description
ERR_CARD_NOT_FOUND	The card could not be detected, operation is not possible.
ERR_INVALID_BLOCK_RANGE	An invalid block range was selected.
ERR_AUTHENTICATION_FAILED	Authentication failed, authorization was not successful.
ERR_BLOCK_READ_FAILED	The specified blocks could not be read.

Summary

This command is used to read data from specific blocks of RFID cards.
Requires authentication (KeyA, KeyB, KeyOptional KeyA, KeyB Optional).
If the block spacing is specified, all blocks between the start and end blocks are read.
Upon successful read, 16 bytes of data for each block are returned in response.
Conditions such as authentication failure, invalid block gap, or card detection failure can cause transaction failure.

This command plays an important role in managing data reads on RFID cards and is used in applications that require secure access.

7.3.3.2.1 RFID Card Block Read Command Example – 1

This example demonstrates the process of reading data from **block 0x0F** of an RFID card. Authentication is performed using **KeyA**, and the retrieved data is returned in **HEX** format.

Sent Message Format

Command Sent (Command to Read Data from RFID Card Blocks)

Command Sent (Command to Read Data from RFID Card Blocks)								
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	KEY_TYPE	AUTH_TYPE	START_ BLOCK	END_ BLOCK	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x08	0x01	0x02	0x0F	0x0F	0x79	0x07

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Specifies the address of the device receiving the command.
- COMMAND_TYPE (0x08) → Defines the RFID card block read command.
- KEY_TYPE (0x01) → Indicates that authentication will be performed using **KeyA**.
- AUTH_TYPE (0x02) → Specifies the **PICC_AUTHENT1B** authentication method.
- START_BLOCK (0x0F) → Specifies the first block address to be read.
- END_BLOCK (0x0F) → Specifies the last block address to be read (single block read).
- CRC_HIGH (0x79) & CRC_LOW (0x07) → Error control mechanism used to ensure data integrity.

Received Message Format

Response (RFID Card Block Read Response - Block 0x0F)

Response (RFID Card Block Read Response - Block 0x0F)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Block 0x0F - 16 Bytes)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x10	0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0x07 0x80 0x69 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF	0xBD	0x24

Field Descriptions

- HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
- MESSAGE_TYPE (0x10) → Represents the message returned from the RFID card block read operation.
- DATA → Contains **16 bytes of data** read from block **0x0F**.
- CRC_HIGH (0xBD) & CRC_LOW (0x24) → Error control mechanism used to verify data integrity.

7.3.3.2.2 RFID Card Block Read Command Example – 2

This example demonstrates the process of reading data from **block 0x01** of an RFID card.
Authentication is performed using **KeyA**, and the retrieved data is returned in **HEX** format

Sent Message Format

Command Sent (Command to Read Data from RFID Card Blocks)

Command Sent (Command to Read Data from RFID Card Blocks)								
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	KEY _TYPE	AUTH _TYPE	START _BLOCK	END _BLOCK	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x08	0x01	0x02	0x01	0x01	0xBB	0xC6

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Specifies the address of the device receiving the command.
- COMMAND_TYPE (0x08) → Defines the RFID card block read command.
- KEY_TYPE (0x01) → Indicates that authentication will be performed using **KeyA**.
- AUTH_TYPE (0x02) → Specifies the **PICC_AUTHENT1B** authentication method.
- START_BLOCK (0x01) → Specifies the first block address to be read.
- END_BLOCK (0x01) → Specifies the last block address to be read (single block read).
- CRC_HIGH (0xBB) & CRC_LOW (0xC6) → Error control mechanism used to ensure data integrity.

Received Message Format

Response (RFID Card Block Read Response - Block 0x01)

Response (RFID Card Block Read Response - Block 0x01)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Block 0x01 - 16 Bytes)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x10	0x52 0x46 0x49 0x44 0x41 0x58 0x20 0x44 0x65 0x76 0x69 0x63 0x65 0x73 0x00 0x00	0x97	0xF3

Field Descriptions

- HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
- MESSAGE_TYPE (0x10) → Represents the message returned from the RFID card block read operation.
- DATA → Contains **16 bytes of data** read from block **0x01**.

Read Block 0x01 Contents

Byte Index	HEX Value	ASCII Equivalent
0	0x52	R
1	0x46	F
2	0x49	I
3	0x44	D
4	0x41	A
5	0x58	X
6	0x20	(Boşluk)
7	0x44	D
8	0x65	e
9	0x76	v
10	0x69	i
11	0x63	c
12	0x65	e
13	0x73	s
14	0x00	NULL
15	0x00	NULL

Additional Notes

- **First 14 bytes** *(0x52 0x46 0x49 0x44 0x41 0x58 0x20 0x44 0x65 0x76 0x69 0x63 0x65 0x73)* → Represents the **ASCII text "RFIDAX Devices"**.
- **Last 2 bytes (0x00 0x00)** → Represents empty data or block termination.

Summary

- In this example, **16 bytes of data** were successfully read from **block 0x01** of the RFID card.
- The device completed the read operation and returned a **16-byte block containing the ASCII text "RFIDAX Devices"**.
- **CRC values (0x97 0xF3)** are used to verify the correctness of the message.
- This message format serves as an essential reference for understanding **RFID block read operations**.

Important Note

In RFIDAX devices, read and write operations are only performed on valid blocks. Special sector blocks are protected from unintended modifications. Writing to these blocks has been intentionally disabled to prevent data corruption or malfunction of RFID cards.

7.3.3.2.3 RFID Card Block Read Command Example – 3

This example demonstrates the process of reading data from a **specified block range** on an RFID card. Authentication is performed using **KeyA, KeyB, Optional_KeyA, or Optional_KeyB**, and data is retrieved in **HEX format** based on the specified start and end block addresses.

Sent Message Format

Command Sent (Command to Read Data from RFID Card Blocks)

Command Sent (Command to Read Data from RFID Card Blocks)								
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	KEY_TYPE	AUTH_TYPE	START_BLOCK	END_BLOCK	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x08	0x01	0x02	0x10	0x12	0xB9	0xF7

Received Message Format

Response (RFID Card Block Read Response)

The following messages return the data read from the specified block range.

HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Block Contents)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x10	0x46 0x6F 0x72 0x20 0x69 0x6E 0x66 0x6F 0x72 0x6D 0x61 0x74 0x69 0x6F 0x6E 0x2C	0xEA	0xF5
0xAA	0x01	0x10	0x20 0x76 0x69 0x73 0x69 0x74 0x20 0x77 0x77 0x77 0x2E 0x72 0x66 0x69 0x64 0x61	0xD6	0xEA
0xAA	0x01	0x10	0x78 0x2E 0x63 0x6F 0x6D 0x2E 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	0xD5	0x5A
0xAA	0x01	0x10	0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0x07 0x80 0x69 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF	0xBD	0x24

Read Block Contents

The following table provides a detailed breakdown of the data read from each block.

Block Address	HEX Values	ASCII Equivalent
0x10	0x46 0x6F 0x72 0x20 0x69 0x6E 0x66 0x6F 0x72 0x6D 0x61 0x74 0x69 0x6F 0x6E 0x2C	For information,
0x11	0x20 0x76 0x69 0x73 0x69 0x74 0x20 0x77 0x77 0x77 0x2E 0x72 0x66 0x69 0x64 0x61	visit www.rfida
0x12	0x78 0x2E 0x63 0x6F 0x6D 0x2E 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	x.com.

Field Descriptions

- HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
- MESSAGE_TYPE (0x10) → Indicates that this is an RFID block read message.
- DATA → Contains the **read block data** in 16-byte segments.
- CRC_HIGH & CRC_LOW → Error control mechanism used to ensure data integrity.

Summary

- In this example, **data was successfully read** from **block range 0x10 - 0x12** of the RFID card.
- The retrieved ASCII text is **"For information, visit www.rfidax.com."**
- Each block contains **16 bytes** of data, and the message is **split across multiple blocks**.
- **CRC values** are included to ensure message integrity and data validation.
- This message format serves as an essential reference for understanding **RFID block read operations** in **multi-block read scenarios**.

7.4 System Control Commands

In RFIDAX devices, system control commands are used to perform operations related to hardware and software management. These commands cover essential functions such as querying the device's operational status, retrieving version information, restarting the system, and detecting errors.
Through these commands, critical information such as hardware and software version numbers, build number, production date, operating mode, and communication protocols can be queried. Additionally, operations such as restarting the device or performing a system reset can be executed via system control commands.
These commands play a significant role in device maintenance, system updates, and debugging. Users and developers can use these commands to obtain information about the system's status, verify specific configurations, and restart the device when necessary.

7.4.1 Purpose of System Control Commands

System control commands are used for the following key purposes:

1. Querying Device Hardware and Software Information
   - Retrieving hardware and software version details.
   - Querying the build number and production date.
2. System Restart and Reset Operations
   - Restarting the device at the software level.
   - Restoring the device to factory settings when required.
3. Viewing Operating Modes and Configurations
   - Checking the current communication mode (USB, RS485, etc.).
   - Verifying whether data integrity checks (CRC) are enabled.
   - Confirming the operating mode (Basic, Advanced, Enterprise, Custom).

7.4.2 System Control Commands and List

Below is a list of fundamental system control commands used in the RFIDAX system:

Command Code	Command Name	Description
0x0A	Software Reset Command	Restarts the device at the software level. This process does not restore the device to factory settings but clears all temporary memory and restarts the system.
0x0D	Retrieve Device Version Information Command	Returns the device's hardware and software version details, build number, and production date. This command is used to learn the current system configuratio

Table 35 Syste
m Control Commands

These commands are crucial for general device management and maintenance processes.

7.4.2.1 Software Reset Command

The **Software Reset Command (0x0A)** allows the device to restart at the software level by clearing temporary system data.

This process does not require a hardware reset; instead, it clears memory and restarts the operating cycle. However, it does not restore factory settings, and permanent data remains unaffected.

This command is required because:

Software reset plays a critical role in debugging RFID systems, maintaining system stability, and restarting the device after software updates.

Physically resetting the device is not always feasible, especially in remote systems or embedded environments where remotely managed reset mechanisms are required.

This command is useful in the following scenarios:

- **When memory or processor load is excessive, ensuring smooth operation by clearing memory.**
- **After a software update or configuration change, applying new settings by restarting the system.**
- **In case of an unexpected error or system freeze, performing a software-level restart without requiring a hardware reset.**
- **During testing and development, resetting the system before loading new firmware.**

By utilizing this command, the system can operate stably without manual intervention, ensuring uninterrupted functionality.

Sent Message Format

Command Sent (Software Reset Command for the Device)					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0A	0xFF	0xC0	0x16

Field Descriptions

- **HEADER (0xAA):** Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** Specifies the address of the device receiving the command.
- **COMMAND_TYPE (0x0A):** Initiates the software reset process.
- **SUB_COMMAND_TYPE (0xFF):** Sub-command indicating the reset type.
- **CRC_HIGH (0xC0) & CRC_LOW (0x16):** Error control mechanism ensuring data integrity.

When this command is successfully processed, the device resets itself and restarts.

If the command is invalid or the system reset process fails, the device may return an error code in response.

If an error occurs, please refer to "6.4.1 General System Status and Errors."

System Response Message

After a reset operation, system setup, or any reconfiguration process, the following **ASCII** message is returned. This message is in **ASCII format**:

Advanced RFID System Initialization (Example Response)
[Communication Mode]: Mobile (Type-C)
[Data Integrity Check]: Enabled (CRC Active)
[Operation Mode]: Advanced
[Hardware Version]: 1.2
[Software Version]: 3.2
[Build Number]: 75
[Build Date]: 202501

This message confirms that the system has successfully restarted and initialized with the specified configurations.

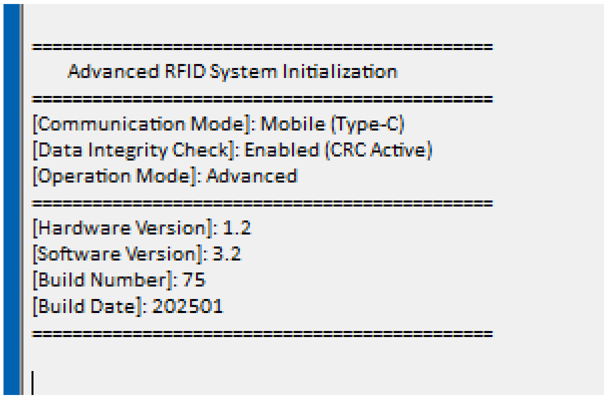

Figure 3 ASCII message after reset (Example)

7.4.2.2 Device Version Information Retrieval Command

The **Device_Version_Info** command is a critical function used to determine a device's **hardware and software version details, build number, manufacturing date, and current operating modes**. This command allows the system to read the **latest version information** of the device, which plays an essential role in **software compatibility, updates, and debugging processes**.
For **system developers and maintenance teams**, the **Device_Version_Info** command provides the following advantages:

1. **Compatibility Check:**
   Verifies the **matching of hardware and software versions**.
   Ensures proper operation of devices with **different versions**.
2. **Debugging and Support:**
   The support team can **quickly identify error sources** using version details.
   Can be used to **track previous updates** applied to the device.
3. **Update Management:**
   Determines **which devices require updates**.
   Ensures **version control** after updates.
4. **Validation of Device Configurations:**
   The **current communication mode (CommMode)**, **CRC check status (CRCMode)**, and **operation mode (OperationMode)** can be read to verify the device configuration.

These details are **crucial in large-scale systems** to properly manage devices with different versions.
Device_Version_Info should be used regularly to **maintain system stability and prevent potential compatibility issues**.

7.4.2 2.1 General Message Structure of Device Version Information Retrieval Command

The message structure used to retrieve device version information consists of two main components:
1. Sent Command:
A **request message** sent to the device to retrieve version details.
Contains **SUB_COMMAND_TYPE** information but **does not include any data content (DATA)**.
CRC calculation is included to **verify message integrity**.
2. Received Response:
A **response message** containing the retrieved version details from the device.
Includes **hardware and software version, build number, manufacturing date, and current operating modes**.
CRC value is included to ensure **message integrity**.
Sent Message Format

Command Sent (Device Version Information Retrieval Command)					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	SUB_COMMAND_TYPE	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0A	0xFF	XX	XX

Field Descriptions:
HEADER (0xAA): Identifies the start of the command. It is a fixed value.
DEVICE_ADDRESS (0x01): Specifies the address of the device receiving the command.
COMMAND_TYPE (0x0A): Command for requesting **device version information**.
SUB_COMMAND_TYPE (0xFF): Sub-command for **reading device version information**.
CRC_HIGH & CRC_LOW: Error control mechanism used to ensure **data integrity**.

Response Message Format

Response Message ((Device Version Information Response))											
HEADER	DEVICE _ADDRESS	MESSAGE_TYPE	Hardware Ver.	Software Ver.	Build No.	Build Date	Comm Mode	CRC Mode	Operation Mode	CRC_HIGH	CRC_LOW
0xAA	0x01	0x02	0xXX 0xXX	0xXX 0xXX	0xXX 0xXX 0xXX 0xXX	0xXX 0xXX 0xXX 0xXX 0xXX 0xXX	0xXX	0xXX	0xXX	0xXX	0xXX

Response Field Descriptions:

HEADER (0xAA): Identifies the start of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01): Indicates the device that sends the response.
MESSAGE_TYPE (0x02): Indicates that the device version information is being sent.
Hardware Version (0xXX 0xXX):

- **Major:** Primary version number of the hardware.
- **Minor:** Secondary version number of the hardware.

Software Version (0xXX 0xXX):

- **Major:** Primary version number of the software.
- **Minor:** Secondary version number of the software.

Build Number (0xXX 0xXX 0xXX 0xXX):

The current software build number of the device.

Build Date (0xXX 0xXX 0xXX 0xXX 0xXX 0xXX):

The **manufacturing date** or **latest firmware update date** of the device.

Communication Mode (0xXX): The current communication mode. See **6.3 Communication Types of Devices** for more details.
CRC Mode (0xXX): The data integrity check status (CRC enabled or disabled). See **6.2 Data Integrity Modes and Descriptions** for more details.
Operation Mode (0xXX): The current operation mode of the device. See **6.1 Device Modes and Descriptions** for more details.
CRC_HIGH & CRC_LOW: Error control mechanism used to **ensure data integrity**

7.4.2 2.1 .1 Example of Device Version Information Retrieval Command Response - 1

AA 01 02 00 01 00 02 00 03 00 02 00 00 00 4B 32 30 32 35 30 31 03 01 01 00 F7 EC

Decoded Response Message:											
HEADER	DEVICE _ADDRESS	MESSAGE _TYPE	Hardware Ver.	Software Ver.	Build No.	Build Date	Comm Mode	CRCMode	Operation Mode	CRC_HIGH	CRC_LOW
0xAA	0x01	0x02	0x00 0x01 0x00 0x02	0x00 0x03 0x00 0x02	0x00 0x00 0x00 0x4B	0x32 0x30 0x32 0x35 0x30 0x31	0x03	0x01	0x01	0xF7	0xEC

Response Message Explanation

BYTE INDEX	HEX VALUE	DESCRIPTION
0	0xAA	**HEADER:** Identifies the start of the response message. It is a fixed value.
1	0x01	**DEVICE_ADDRESS:** Indicates the device that sends the response.
2	0x02	**MESSAGE_TYPE:** Specifies that this is a device version information response.
3-4	0x00 0x01	**Hardware Version:** Device hardware version (1.2).
5-6	0x00 0x02	**Software Version:** Device software version (3.2).
7-10	0x00 0x00 0x00 0x4B	**Build Number:** The device's build number (75).
11-16	0x32 0x30 0x32 0x35 0x30 0x31	**Build Date (YYYYM Format):** 202501 (January 2025).
17	0x03	**Communication Mode:** Device's communication mode (COMM_TYPE_C).
18	0x01	**CRC Mode:** Data integrity check is enabled (0x01).
19	0x01	**Operation Mode:** Advanced Mode (MODE_ADVANCED) is active.
20	0xF7	**CRC_HIGH:** First byte of the error control mechanism ensuring data integrity.
21	0xEC	**CRC_LOW:** Second byte of the error control mechanism ensuring data integrity.

Summary

This message provides **critical information** about the device's **hardware and software versions, build number, manufacturing date, and operating modes.**

- **Hardware and Software Versions:** 1.2 (HW) and 3.2 (SW).
- **Build Number:** 75.
- **Manufacturing Date:** January 2025 (202501).
- **Communication Mode: 0x03 → COMM_TYPE_C.**
- **CRC Verification: 0x01 → Enabled, ensuring message integrity.**
- **Operating Mode: 0x01 → Advanced Mode (MODE_ADVANCED).**

This structure is designed to **retrieve up-to-date version information, verify software compatibility, and facilitate troubleshooting procedures.**

7.5 Memory Management Commands

In RFIDAX devices, **Memory Management Commands** are used to manage the device's **security keys (OptionalKeyA and OptionalKeyB), identity information (ID), and specific memory sections**. These commands function according to predefined security levels established by authorized users of the device.
Memory management is a specialized system within the device that **stores and updates authentication keys securely**. Additionally, **ID management commands** allow users to **read, modify, or reset** the RFID device's unique identity information.
These commands are **designed for advanced access control and security operations**, ensuring that the RFID device is managed correctly and securely by users, system administrators, or security protocols.

7.5.1 Purpose of Memory Management Command

Memory management commands serve the following objectives:

- **Managing Security Keys**
  - Storing and reading **OptionalKeyA** and **OptionalKeyB** values
  - Managing **authentication processes**
- **Identity (ID) Operations**
  - Updating the **device's unique identity number (ID)**
  - Reading and resetting existing **ID information**
  - Listing all registered **IDs** stored in the device

These commands play a **critical role** in **enhancing data security, access control, and authentication systems** within the device.

Command Code	Command Name	Description
0x0B	OptionalKeyA and OptionalKeyB Management	Storing and reading **security keys**
0x0C	ID Management	Unique identity (ID) operations

Table 36 Memory Management Commands

0x0B - OptionalKeyA and OptionalKeyB Operations

Sub Command Code	Operation
0x0B / 0x01	Write OptionalKeyA
0x0B / 0x02	Write OptionalKeyB
0x0B / 0x03	Read OptionalKeyA
0x0B / 0x04	Read OptionalKeyB

Table 37 OptionalKeyA and OptionalKeyB Operations

0x0C - ID Operations

Sub Command Code	Operation
0x0C / 0x01	Update ID
0x0C / 0x02	Read ID
0x0C / 0x03	Reset ID
0x0C / 0x04	Display All IDs

Table 38 ID Operations

7.5.2 OptionalKeyA and OptionalKeyB Operations

In **RFIDAX** devices, **OptionalKeyA and OptionalKeyB** are user-defined **authentication keys**. Unlike standard **KeyA and KeyB**, these keys **can be written and read**, providing enhanced security and customizable access control.

7.5.2.1 Purpose of OptionalKeyA and OptionalKeyB Commands

Optional Key operations are used to manage the **secure access** of RFID cards. The following functions can be performed using these commands:

Updating (Writing) OptionalKeyA and OptionalKeyB

- A new **OptionalKeyA or OptionalKeyB** value can be set and stored in **Flash Memory**.
- Authorized users can **update** the authentication keys when necessary.

Reading OptionalKeyA and OptionalKeyB

- Previously stored **OptionalKeyA or OptionalKeyB** values can be **retrieved**.
- The **current authentication settings** of the device can be verified.

7.7.5.2.2 Sent Message Format

Command Sent (Command to Read or Write OptionalKeyA / OptionalKeyB)						
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	KEY_TYPE	DATA (16 Bytes - Optional)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0B	0xXX	(Only for Write Commands)	0xXX	0xXX

Field Descriptions

- **HEADER (0xAA)** → Identifies the start of the command. This is a fixed value.
- **DEVICE_ADDRESS (0x01)** → Specifies the device to which the command is sent.
- **COMMAND_TYPE (0x0B)** → Indicates that the operation relates to **OptionalKeyA or OptionalKeyB**.
- **KEY_TYPE (0xXX)** → Specifies the operation type:
  - **0x01** → Write **OptionalKeyA**
  - **0x02** → Write **OptionalKeyB**
  - **0x03** → Read **OptionalKeyA**
  - **0x04** → Read **OptionalKeyB**
- **DATA (16 Bytes - Optional)** →
  - In **write commands**, this field contains the **new key data**.
  - In **read commands**, this field **is not used**.
- **CRC_HIGH (XX) & CRC_LOW (XX) (Optional)** → Used as an **error control mechanism** to ensure data integrity.

7.5.2.3 Received Message Format (After Writing a Key)

Response (Optional Key Write Response)					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x00	0x00	0xXX	0xXX

Field Descriptions

- **HEADER (0xAA)** → Identifies the start of the response message. This is a fixed value.
- **DEVICE_ADDRESS (0x01)** → Indicates the device sending the response.
- **STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00)** →
  - **0x00 0x00** → **Write operation was successful.**
  - **Any other error code** → The write operation **failed**. **Refer to section "6.4.1 General System Status and Errors" for more details.**
- **CRC_HIGH (XX) & CRC_LOW (XX) (Optional)** → Used as an **error control mechanism** to ensure data integrity.

7.5.2.4 Received Message Format (After Read Operation)

Response (Optional Key Read Response)

HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (16 Bytes - Key Value)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x11 (KeyA) / 12 (KeyB)	16 Byte Key Data	0xXX	0xXX

Field Descriptions

HEADER (0xAA) → Identifies the start of the response message. This is a fixed value.
DEVICE_ADDRESS (0x01) → Indicates the device sending the response.
MESSAGE_TYPE:
- 0x11 → Indicates the **OptionalKeyA** value.
- 0x12 → Indicates the **OptionalKeyB** value.

DATA (16 Bytes) → Contains the retrieved **OptionalKeyA or OptionalKeyB** value.
CRC_HIGH (XX) & CRC_LOW (XX) (Optional) → Used as an **error control mechanism** to ensure data integrity.

7.5.2.5 OptionalKeyA Write Command Example

This example illustrates the structure of the command used to update the **OptionalKeyA** value and the process flow.
Sent Message Format

Command Sent (Command to Write OptionalKeyA to RFID System)						
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	KEY_TYPE	DATA (16 Bytes - New Key Value)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0B	0x01	0x1A 0x2A 0x3A 0x4A 0x5A 0x6A	0x4D	0x06

Field Descriptions

HEADER (0xAA) → Identifies the start of the command. This is a fixed value.
DEVICE_ADDRESS (0x01) → Specifies the device to which the command is sent.
COMMAND_TYPE (0x0B) → Specifies the **OptionalKey** operation command.
KEY_TYPE (0x01) → Indicates that the operation is **writing OptionalKeyA**.
DATA (16 Bytes - New Key Value) → Contains the new **OptionalKeyA** value (16 bytes).
- Example: **0x1A 0x2A 0x3A 0x4A 0x5A 0x6A**

CRC_HIGH (0x4D) & CRC_LOW (0x06) (Optional) → Used as an **error control mechanism** to ensure data integrity.

Received Message Format

Response (OptionalKeyA Write Response)					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xBB	0x01	0x00	0x00	0x5C	0x3E

Field Descriptions

HEADER (0xBB) → Identifies the start of the response message. This is a fixed value.
DEVICE_ADDRESS (0x01) → Indicates the device sending the response.
STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00) →
- 0x00 0x00 → Write operation was successful.
- Any other error code → Write operation failed.
- Refer to section "6.4.1 General System Status and Errors" for details.

CRC_HIGH (0x5C) & CRC_LOW (0x3E) (Optional) → Used as an **error control mechanism** to ensure data integrity.

7.5.2.6 OptionalKeyA Read Command Example

This example demonstrates the structure and process of the command used to **read the OptionalKeyA value**. OptionalKeyA is a special key used in **RFID authentication processes**. Users can retrieve the current OptionalKeyA value stored in the device using this command.

Sent Message Format

Command Sent (Command to Read OptionalKeyA from RFID System)					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	KEY_TYPE	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0B	0x03	0xDD	0XB4

Field Descriptions

HEADER (0xAA) → Identifies the start of the command. This is a fixed value.
DEVICE_ADDRESS (0x01) → Specifies the device to which the command is sent.
COMMAND_TYPE (0x0B) → Specifies the **OptionalKey operations command**.
KEY_TYPE (0x03) → Indicates that the operation is **reading OptionalKeyA**.
CRC_HIGH (0xDD) & CRC_LOW (0xB4) (Optional) → Used as an **error control mechanism** to ensure data integrity.

Received Message Format

Response (OptionalKeyA Read Response)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (OptionalKeyA - 6 Bytes)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x11	0x1A 0x2A 0x3A 0x4A 0x5A 0x6A	0x47	0x47

Field Descriptions

HEADER (0xAA) → Identifies the start of the response message. This is a fixed value.
DEVICE_ADDRESS (0x01) → Indicates the device sending the response.
MESSAGE_TYPE (0x11) → Indicates that the **OptionalKeyA read message** is being returned.
DATA (OptionalKeyA - 6 Bytes) → Contains the retrieved **OptionalKeyA value**.
Example: **0x1A 0x2A 0x3A 0x4A 0x5A 0x6A**
CRC_HIGH (0x47) & CRC_LOW (0x47) (Optional) → Used as an **error control mechanism** to ensure data integrity.

Steps in This Process

1. The user **sends a request** to read the **OptionalKeyA value** from the RFID device.
2. The device retrieves the **stored OptionalKeyA value** from its memory.
3. The **retrieved OptionalKeyA value** is returned in a **6-byte data field** within the response message.
4. To ensure message integrity, the **CRC values** are included in the response.

This command allows **authorized users** to retrieve the **current OptionalKeyA** for authentication and security validation.

7.5.2.7 OptionalKeyB Write Command Example

This example demonstrates the structure and process of the command used to **update the OptionalKeyB value**. OptionalKeyB is a special key used in **RFID authentication processes**, and users can modify it using this command.

Sent Message Format

Command Sent (Command to Write OptionalKeyB to RFID System)						
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	KEY_TYPE	DATA (OptionalKeyB - 6 Bytes)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0B	0x02	0xA1 0xB2 0x99 0xD1 0xE1 0xF1	0x80	0x6A

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. This is a fixed value.
- DEVICE_ADDRESS (0x01) → Specifies the device to which the command is sent.
- COMMAND_TYPE (0x0B) → Specifies the **OptionalKey operations command**.
- KEY_TYPE (0x02) → Indicates that the operation is **writing OptionalKeyB**.
- DATA (OptionalKeyB - 6 Bytes) → The new **OptionalKeyB value** provided by the user.
  - Example: **0xA1 0xB2 0x99 0xD1 0xE1 0xF1**
- CRC_HIGH (0x80) & CRC_LOW (0x6A) (Optional) → Used as an **error control mechanism** to ensure data integrity.

Received Message Format

Response (OptionalKeyB Write Response)					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xBB	0x01	0x00	0x00	0x5C	0x3E

Field Descriptions

- HEADER (0xBB) → Identifies the start of the response message. This is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device sending the response.
- STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00) →
  - 0x00 0x00 → Indicates that the **write operation was successful**.
- CRC_HIGH (0x5C) & CRC_LOW (0x3E) (Optional) → Used as an **error control mechanism** to ensure data integrity.

7.5.2.8 OptionalKeyB Read Command Example

This example demonstrates the **structure and process** of the command used to **read the OptionalKeyB value**. OptionalKeyB is a **special key used in RFID authentication processes**. Users can retrieve the currently stored **OptionalKeyB value** from the device using this command.

Sent Message Format

Command Sent (Command to Read OptionalKeyB from RFID System)					
HEADER	DEVICE_ADDRESS	COMMAND_TYPE	KEY_TYPE	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0B	0x04	0xAD	0x53

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. This is a fixed value.
- DEVICE_ADDRESS (0x01) → Specifies the device to which the command is sent.
- COMMAND_TYPE (0x0B) → Specifies the **OptionalKey operations command**.
- KEY_TYPE (0x04) → Indicates that the operation is **reading OptionalKeyB**.
- CRC_HIGH (0xAD) & CRC_LOW (0x53) (Optional) → Used as an **error control mechanism** to ensure data integrity.

Received Message Format

Response (OptionalKeyB Read Response)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (OptionalKeyB - 6 Bytes)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x12	0xA1 0xB2 0x99 0xD1 0xE1 0xF1	0x8A	0x2B

Field Descriptions

- HEADER (0xAA) → Identifies the start of the response message. This is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device sending the response.
- MESSAGE_TYPE (0x12) → Indicates that the **returned value corresponds to OptionalKeyB**.
- DATA (0xA1 0xB2 0x99 0xD1 0xE1 0xF1) → The stored **OptionalKeyB value**.
- CRC_HIGH (0x8A) & CRC_LOW (0x2B) (Optional) → Used as an **error control mechanism** to ensure data integrity.

7.5.2.9 Summary

- **OptionalKeyA and OptionalKeyB** are **special key values** used for **advanced authentication and security operations**.
- The **0x0B command group** manages **writing and reading** these keys.
- **After a successful write operation**, the device **returns a success or error response**.
- **After a read operation**, the device **returns the stored value** of the requested key.
- This section provides a **detailed explanation of OptionalKeyA and OptionalKeyB operations**, ensuring secure authentication management.

7.5.2 ID Operations

In RFIDAX devices, **ID Operations Commands** are used to manage the unique identification (ID) data of the device. ID management is a crucial component in determining device identity, controlling user access permissions, and guiding authentication processes within the system.
In RFID systems, **ID information represents the unique identity of a device or specific users**. These details are stored in the device's memory and can be **updated, read, reset, or listed as needed**.

7.5.3.1Purposes of ID Operations

The **ID management commands** are used to perform the following functions:
ID Update

The existing **ID information of the device can be changed**, or a **new ID can be assigned**.
Authorized users can update predefined ID numbers in the system.
ID numbers are generally used for **access control and device authorization**.

ID Read

The **current ID information** can be retrieved.
Specific **ID data stored in the device** can be accessed.
With proper authorization controls, **identity details can be viewed within the system**.

ID Reset

A specific **ID can be completely erased**, removing previous information.
ID reset operations are useful for clearing misconfigured or invalid IDs.

List All IDs

This command retrieves **a complete list of all registered IDs** stored in the device.
System administrators can review and manage all stored identity data.

7.5.3.2 Importance of Memory Management and ID Operations

ID operations in the RFIDAX system play a critical role in identification, authorization, and security processes.
Authorized users can manage ID-based access controls and strengthen authentication mechanisms with these commands.
ID update and read operations allow the device to be uniquely identified for specific users.
Proper ID management is essential for advanced authentication mechanisms.

7.5.3.3 Sent Message Format

ID management commands are used to **update, read, reset, or list all IDs** stored in the device.
These commands are sent in a specific format that includes **ID length and ID index values**.
A maximum of **10 IDs** can be managed within the device's memory.

Command Sent (Command for ID Management Operations)								
HEADER	DEVICE_ ADDRESS	COMMAND _TYPE	SUB_COMMAND _TYPE	ID_L ENGTH	ID _INDEX	DATA (Variable)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0C	0xXX	0xXX	0xXX	(Only for Write)	0xXX	0xXX

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Specifies the address of the device receiving the command.
- COMMAND_TYPE (0x0C) → Represents the **main command type** for ID operations.
- SUB_COMMAND_TYPE (0xXX) → Defines the operation to be performed:
  - 0x01 → Update ID
  - 0x02 → Read ID
  - 0x03 → Reset ID
  - 0x04 → Show All IDs
- ID_LENGTH (0xXX) → Specifies the length of the ID to be updated or read.
- ID_INDEX (0xXX) → Specifies the index number of the ID to be updated or read.
- DATA (Variable - Only for Write ID) → Used only in **ID Update** commands and contains the new ID value.
- CRC_HIGH (XX) & CRC_LOW (XX) (Optional) → Error control mechanism ensuring data integrity.

7.5.3.4 Received Message Format (ID Update, Read, and Reset Response)

The response message from the device contains the result of the operation and, if applicable, the retrieved ID data

Response (ID Management Response)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Variable - Only for Read)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0xXX	ID Data (Optional)	0xXX	0xXX

Field Descriptions

- HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
- MESSAGE_TYPE (0xXX) → Defines the result of the ID operation:
  - 0x20 + ID (Hex Value) → ID Successfully Updated
  - 0x50 + ID (Hex Value) → ID Read Result
  - 0x8C + ID (Hex Value) → ID Successfully Reset
  - 0xC8 → All IDs Listed
- DATA (Variable - Only for Read ID & Show All IDs) → Contains the retrieved ID data.
- CRC_HIGH (XX) & CRC_LOW (XX) (Optional) → Error control mechanism ensuring data integrity.

7.5.3.5 ID Update Command Example

This example demonstrates the **structure and process** of the **ID update command** used in the **RFID system**.
This process is used to **change an existing ID** or **assign a new ID**.

1st Example: ID 0 Update Process

This operation is used to **update the value of the ID at index 0**.

Sent Message Format

Command Sent (Command to Update ID 0 in RFID System)								
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	SUB_COMMAND _TYPE	ID_ LENGT H	ID _INDEX	DATA (New ID Value)	CRC _HIGH (Optional)	CRC _LOW (Optional)
0xAA	0x01	0x0C	0x01	0x04	0x00	0x12 0x34 0x56 0x78	0x58	0xB9

Field Descriptions

HEADER (0xAA) → Identifies the start of the command. It is a fixed value.
DEVICE_ADDRESS (0x01) → Specifies the address of the device receiving the command.
COMMAND_TYPE (0x0C) → Main command type for ID operations.
SUB_COMMAND_TYPE (0x01) → Specifies that this is an **ID update operation**.
ID_LENGTH (0x04) → Indicates the length of the ID to be updated (**4 bytes**).
ID_INDEX (0x00) → Specifies the index of the ID to be updated.
DATA (0x12 0x34 0x56 0x78) → The new ID value.
CRC_HIGH (0x58) & CRC_LOW (0xB9) → **Error control mechanism** ensuring data integrity.

Received Message Format

Response (ID Update Response for ID 0)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Updated ID 0)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x20	0x12 0x34 0x56 0x78	0x3E	0xB7

Field Descriptions

HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
MESSAGE_TYPE (0x20) → **Indicates the ID update operation**. For more details, see (6.7.2.1.2 Flash Memory ID Update Messages and Message Type Information).
DATA (0x12 0x34 0x56 0x78) → Updated ID value.
CRC_HIGH (0x3E) & CRC_LOW (0xB7) → **Error control mechanism** ensuring data integrity.

Note:

MESSAGE_TYPE value varies based on the ID number. It is calculated as **0x20 + ID number**.

2nd Example: ID 9 Update Process

This operation is used to **update the value of the ID at index 9**.

Sent Message Format

Command Sent (Command to Update ID 9 in RFID System)								
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	SUB_ COMMAND _TYPE	ID _LENGTH	ID _INDEX	DATA (New ID Value)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0C	0x01	0x04	0x09	0x6F 0x6E 0x75 0x72	0xE0	0xC6

Field Descriptions

- **HEADER (0xAA)** → Identifies the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01)** → Specifies the address of the device receiving the command.
- **COMMAND_TYPE (0x0C)** → Main command type for ID operations.
- **SUB_COMMAND_TYPE (0x01)** → Specifies that this is an **ID update operation**.
- **ID_LENGTH (0x04)** → Indicates the length of the ID to be updated (**4 bytes**).
- **ID_INDEX (0x09)** → Specifies the index of the ID to be updated.
- **DATA (0x6F 0x6E 0x75 0x72)** → The new ID value.
- **CRC_HIGH (0xE0) & CRC_LOW (0xC6)** → **Error control mechanism** ensuring data integrity.

Received Message Format

Response (ID Update Response for ID 9)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Updated ID 9)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x29	0x6F 0x6E 0x75 0x72	0x86	0xC8

Field Descriptions

- **HEADER (0xAA)** → Identifies the start of the response message. It is a fixed value.
- **DEVICE_ADDRESS (0x01)** → Indicates the device that sends the response.
- **MESSAGE_TYPE (0x29)** → **Indicates the ID update operation** (MESSAGE_TYPE = **0x20 + ID_INDEX**).
  - For details, see (6.7.2.1.2 Flash Memory ID Update Messages and Message Type Information).
- **DATA (0x6F 0x6E 0x75 0x72)** → Updated ID value.
- **CRC_HIGH (0x86) & CRC_LOW (0xC8)** → **Error control mechanism** ensuring data integrity

Note:

- MESSAGE_TYPE value is determined by the updated ID number.
  - It is calculated as 0x20 + ID_INDEX.
  - ID_INDEX = 9 → MESSAGE_TYPE = 0x20 + 0x09 = 0x29.
- Each ID update operation returns a response message based on the updated ID number.
- CRC values are included to ensure message integrity and prevent data corruption.

7.5.3.6 ID Read Command Example

This example demonstrates the structure and process of the command used to **read an ID value** from the RFID device.

This operation is used to retrieve an **existing ID** stored in the device.

1st Example: Reading ID 1

This operation is used to **read the value stored at ID index 1**.

Sent Message Format

Command Sent (Command to Read ID 1 from RFID System)							
HEADER	DEVICE_ ADDRESS	COMMAND _TYPE	SUB_COMMAND _TYPE	ID _LENGTH	ID _INDEX	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0C	0x02	0x04	0x01	0x3E	0xDC

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Specifies the address of the device receiving the command.
- COMMAND_TYPE (0x0C) → Main command type for **ID operations**.
- SUB_COMMAND_TYPE (0x02) → Specifies an **ID read operation**.
- ID_LENGTH (0x04) → Indicates the **length of the ID** to be read (**4 bytes**).
- ID_INDEX (0x01) → Specifies the **index of the ID** to be read.
- CRC_HIGH (0x3E) & CRC_LOW (0xDC) → **Error control mechanism** ensuring data integrity.

Received Message Format

Response (ID Read Response for ID 1)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Read ID 1)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x51	0x11 0x22 0x33 0x44	0xE0	0x7F

Field Descriptions

- HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
- MESSAGE_TYPE (0x51) → **Indicates the ID read operation** (MESSAGE_TYPE = 0x50 + ID_INDEX).
  - For details, see (6.7.2.1.3 Flash Memory ID Read Messages and Message Type Information).
- DATA (0x11 0x22 0x33 0x44) → The **retrieved ID value**.
- CRC_HIGH (0xE0) & CRC_LOW (0x7F) → **Error control mechanism** ensuring data integrity.

Note:

- MESSAGE_TYPE value is determined by the ID number being read.
  - It is calculated as 0x50 + ID_INDEX.
  - ID_INDEX = 1 → MESSAGE_TYPE = 0x50 + 0x01 = 0x51.
- Each ID read operation returns a response message based on the requested ID number.
- CRC values are included to ensure message integrity and prevent data corruption.

2nd Example: Reading ID 9

This operation is used to **read the value stored at ID index 9**.

Sent Message Format

Command Sent (Command to Read ID 9 in RFID System)							
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	SUB _COMMAND_TYPE	ID _LENGTH	ID _INDEX	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0C	0x02	0x04	0x09	0xBF	0xD4

Field Descriptions

HEADER (0xAA) → Identifies the start of the command. It is a fixed value.
DEVICE_ADDRESS (0x01) → Specifies the address of the device receiving the command.
COMMAND_TYPE (0x0C) → Main command type for **ID operations**.
SUB_COMMAND_TYPE (0x02) → Specifies an **ID read operation**.
ID_LENGTH (0x04) → Indicates the **length of the ID** to be read (**4 bytes**).
ID_INDEX (0x09) → Specifies the **index of the ID** to be read.
CRC_HIGH (0xBF) & CRC_LOW (0xD4) → **Error control mechanism** ensuring data integrity.

Received Message Format

Response (ID Read Response for ID 9)					
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	DATA (Read ID 9)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x59	0x6F 0x6E 0x75 0x72	0x9B	0x4E

Field Descriptions

HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
MESSAGE_TYPE (0x59) → **Indicates the ID read operation** (MESSAGE_TYPE = 0x50 + ID_INDEX).
CRC_HIGH (0x9B) & CRC_LOW (0x4E) → **Error control mechanism** ensuring data integrity.

Note:

MESSAGE_TYPE value is determined by the ID number being read.
It is calculated as 0x50 + ID_INDEX.
ID_INDEX = 9 → MESSAGE_TYPE = 0x50 + 0x09 = 0x59.
CRC values are included to ensure message integrity and prevent data corruption.

7.5.3.7 ID Reset Command Example

This example demonstrates the command structure and process used to reset a specific ID value in the RFID system.

This operation is used to erase the existing ID information at the specified ID index and reset it to the default value.

1st Example: Resetting ID 1

This operation is used to reset the value stored at **ID index 1**.

Sent Message Format

Command Sent (Command to Reset ID 1 in RFID System)

HEADER	DEVICE _ADDRESS	COMMAND _TYPE	SUB _COMMAND_TYPE	ID _LENGTH	ID_ INDEX	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0C	0x03	0x04	0x01	0x09	0xEC

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Specifies the address of the device receiving the command.
- COMMAND_TYPE (0x0C) → Main command type for **ID operations**.
- SUB_COMMAND_TYPE (0x03) → Specifies an **ID reset operation**.
- ID_LENGTH (0x04) → Indicates the **length of the ID** to be reset (**4 bytes**).
- ID_INDEX (0x01) → Specifies the **index of the ID** to be reset.
- CRC_HIGH (0x09) & CRC_LOW (0xEC) → **Error control mechanism** ensuring data integrity.

Received Message Format

Response (ID Reset Response for ID 1)

HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x8D	0xC5	0xD5

Field Descriptions

- HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
- DEVICE_ADDRESS (0x01) → Indicates the device that sends the response.
- MESSAGE_TYPE (0x8D) → **Message code for the ID reset operation**.
  - The **MESSAGE_TYPE value is generated by adding 1** to the ID's **assigned message code (0x8C)**.
  - **For example:**
    - ID read **message code = 0x8C**
    - Reset operation MESSAGE_TYPE = 0x8C + 1 = 0x8D
- CRC_HIGH (0xC5) & CRC_LOW (0xD5) → **Error control mechanism** ensuring data integrity.

2nd Example: Resetting ID 9

This example demonstrates the **command structure and process** used to reset the **value of ID index 9** in the **RFID system**.

Sent Message Format

Command Sent (Command to Reset ID 9 in the RFID System)							
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	SUB _COMMAND_TYPE	ID _LENGTH	ID _INDEX	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0C	0x03	0x04	0x09	0x88	0xE4

Field Descriptions

- HEADER (0xAA) → Identifies the start of the command. It is a **fixed value**.
- DEVICE_ADDRESS (0x01) → Specifies the **address of the device** receiving the command.
- COMMAND_TYPE (0x0C) → Main command for **ID operations**.
- SUB_COMMAND_TYPE (0x03) → Specifies an **ID reset operation**.
- ID_LENGTH (0x04) → Indicates the **length of the ID** to be reset (**4 bytes**).
- ID_INDEX (0x09) → Specifies the **index of the ID** to be reset.
- CRC_HIGH (0x88) & CRC_LOW (0xE4) → **Error control mechanism** ensuring **data integrity**.

Received Message Format

Response (ID Reset Response for ID 9)				
HEADER	DEVICE_ADDRESS	MESSAGE_TYPE	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x95	0x56	0xEC

Field Descriptions

HEADER (0xAA) → Identifies the start of the response message. It is a **fixed value**.
DEVICE_ADDRESS (0x01) → Indicates the **device that sends the response**.
MESSAGE_TYPE (0x95) → **Message code for the ID reset operation**.
The **MESSAGE_TYPE value is generated by adding 1** to the **assigned message code for ID 9 (0x94)**.
For example:
ID read MESSAGE_TYPE = 0x94
Reset operation MESSAGE_TYPE = 0x94 + 1 = 0x95
CRC_HIGH (0x56) & CRC_LOW (0xEC) → **Error control mechanism** ensuring **data integrity**.

7.5.3.7 Listing All IDs

This command is used to **list all IDs stored in the RFID device**.
It queries all **ID values in the device's memory** and returns them as a response.

Sent Message Format

Command Sent (Command to List All IDs in the RFID System)							
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	SUB _COMMAND_TYPE	ID _LENGTH	ID _INDEX	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0C	0x04	0x04	0x00	0x9C	0x5D

Field Descriptions

HEADER (0xAA) → Identifies the start of the command. It is a **fixed value**.
DEVICE_ADDRESS (0x01) → Specifies the **address of the device** receiving the command.
COMMAND_TYPE (0x0C) → Main command type for **ID operations**.
SUB_COMMAND_TYPE (0x04) → Specifies an **operation to list all IDs**.
ID_LENGTH (0x04) → Specifies the **length of each ID** being listed.
ID_INDEX (0x00) → Defines the **starting index** of the ID listing process.
CRC_HIGH (0x9C) & CRC_LOW (0x5D) → **Error control mechanism** ensuring **data integrity**.

Received Message Format

Response (List All IDs Response)					
HEADER	DEVICE _ADDRESS	MESSAGE _TYPE	DATA (All ID Values)	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0xC8	0x12 0x34 0x56 0x78 0x11 0x22 0x33 0x44 0xAA 0x22 0xCC 0xDD 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xE1 0x2E 0x3D 0x4B 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0x6F 0x6E 0x75 0x72 0x5A 0x00	-	-

Field Descriptions:

HEADER (0xAA) → Identifies the start of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01) → Indicates the address of the device sending the response.
MESSAGE_TYPE (0xC8) → Message code for the "List All IDs" response.
DATA → Contains all the ID values read from the device:
0x12 0x34 0x56 0x78 → ID 0
0x11 0x22 0x33 0x44 → ID 1
0xAA 0x22 0xCC 0xDD → ID 2
0xFF 0xFF 0xFF 0xFF → Empty ID (Not Registered)
0xE1 0x2E 0x3D 0x4B → ID 3
0xFF 0xFF 0xFF 0xFF → Empty ID (Not Registered)
0x6F 0x6E 0x75 0x72 → ID 9
0x5A 0x00 → Data Terminator

Steps Performed in This Process:

1. The user sends a command to the device to list all IDs.
2. The device reads all ID values stored in its memory.
3. The existing ID information is returned in the response message.
4. CRC values are added to ensure message integrity.

This command is used to query all IDs stored in the device's memory.
Stored ID values are returned, while empty ID fields are marked with 0xFF.

7.6 Device Configuration Commands

In RFIDAX devices, **Device Configuration Commands** are used to modify and configure the device's fundamental

operating parameters.
These commands include changing the **Device ID**, setting the **communication protocol**, and adjusting the **baud rate**.
These commands are used to manage and update the device's long-term configuration and make permanent changes in **Flash Memory**.
Only authorized users can use these commands as they directly affect the device's operation.

7.6.1 Purpose of Device Configuration Commands

Device configuration commands are used to perform the following operations:

1. Changing the Device ID

- Modifying the device's unique identification number (**Device ID**).
- Ensuring that each device in a multi-device system has a different **ID**.
- Updating the **default device address** and saving it in **Flash Memory**.

2. Changing the Protocol

- Modifying the **communication protocol** of the device.
- Selecting the appropriate **CRC standard** for communication (e.g., **CRC16, MODBUS, etc.**).
- Storing the new protocol information in **Flash Memory**.

3. Changing the Baud Rate

- Adjusting the **communication speed (baud rate)** of the device.
- Selecting the baud rate from a predefined **baud rate list**.
- Updating and saving the **baud rate** in **Flash Memory**.

These configuration operations allow the device to be compatible with different systems and to be customized based on **user requirements**.

7.6.2 List of Device Configuration Commands

Below is a list of **Device Configuration Commands** and their functions:

Command Code	Sub Command Code	Operation
0x0E	0x01	Change Device ID
0x0E	0x02	Protocol CRC Change
0x0E	0x03	Change Baud Rate

Table 39 Device Configuration Command List

7.6.3 Device ID Change Command

The **Device ID Change Command** is used to modify the device's unique address.

RFIDAX devices use this **Device ID** for communication.
This command ensures that multiple devices within the same system can be distinguished from each other.
The **Device ID change** is stored **in Flash Memory**, and the device will continue operating with the updated address after a restart.
Note: The **maximum Device ID** is **255**. Any value above **255** is not allowed.

7.6.3.1 Purpose of Use

This command is used for the following purposes:

- Changing the communication **Device ID** of the device.
- Assigning **unique addresses** to each RFID device in a multi-device system.
- Updating the **default Device ID**.
- Storing the **new Device ID** in **Flash Memory** to make it permanent.

7.6.3.2 Sent Message Format

Command Sent (Command to Change Device ID in the RFID System)						
HEADER	DEVICE _ADDRESS	COMMAND _TYPE	SUB_ COMMAND_TYPE	NEW _DEVICE_ID	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0E	0x01	0xXX	0xXX	0xXX

Field Descriptions

- **HEADER (0xAA)** → Indicates the start of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01)** → Represents the current address of the device receiving the command.
- **COMMAND_TYPE (0x0E)** → Defines the main command type for device configuration.
- **SUB_COMMAND_TYPE (0x01)** → Specifies the **Device ID change** operation.
- **NEW_DEVICE_ID (0xXX)** → Represents the new **Device ID** to be assigned.
- **CRC_HIGH (XX) & CRC_LOW (XX) (Optional)** → Used as an error-checking mechanism to ensure data integrity.

7.6.3.3 Received Message Format (Device ID Change Response)

If the Device ID change operation is successful, the device will return the following response:

HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0xXX	0x00	0x00	0xXX	0xXX

Field Descriptions

- **HEADER (0xAA)** → Indicates the start of the response message. It is a fixed value.
- **DEVICE_ADDRESS (0xXX)** → Represents the updated **Device ID**.
- **STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00)** →
  - **0x00 0x00** → Indicates the **operation was successful**.
- **CRC_HIGH (XX) & CRC_LOW (XX) (Optional)** → Used as an error-checking mechanism to ensure data integrity.

1st Example: Updating Device ID to 3

This example demonstrates the command sent to change the Device ID address to 0x03 and the response received.

Sent Message (Update command to Device ID 3)

Sent Message (Update command to Device ID 3)						
HEADER	DEVICE_ ADDRESS	COMMAND _TYPE	SUB _COMMAND_TYPE	NEW_DEVICE _ID	CRC_HIGH	CRC_LOW
0xAA	0x01	0x0E	0x01	0x03	0x13	0x21

Received Message (Successfully Updated, New Address 0x03)

Response Message (Successfully Updated, New Address 0x03)					
HEADER	DEVICE_ADDRESS (New)	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH	CRC_LOW
0xBB	0x03	0x00	0x00	0x32	0x5E

Field Descriptions

HEADER (0xAA - Sent / 0xBB - Received) → Indicates the start of the command and response.
DEVICE_ADDRESS (0x01 - Sent / 0x03 - Received) →
In the sent message: The command is sent to the current device address (0x01).
In the received message: The device responds with the new address (0x03).
COMMAND_TYPE (0x0E) → Represents device configuration operations.
SUB_COMMAND_TYPE (0x01) → Specifies the Device ID change operation.
NEW_DEVICE_ID (0x03) → The new assigned Device ID.
STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00) → A value of **0x00 0x00** indicates a successful operation.
CRC_HIGH (0x32) & CRC_LOW (0x5E) → Used for ensuring message integrity.

Steps in This Process

1. The user sends the command to change the device’s Device ID address.
2. The device stores the new Device ID in Flash Memory and updates its system.
3. The response message is returned with the newly updated Device ID (0x03).
4. CRC values are added to ensure the correctness of the message.
5. The device now continues communication with the new Device ID (0x03).

Important Notes

The Device ID change is **permanent**, and the device will continue to operate with the new ID after a restart.
If an **invalid Device ID** is sent, the device may return an **ERR_INVALID_DEVICE_ADDRESS** error.
For relevant error handling, refer to “6.4.5 Advanced Errors and System Management.”
Once successfully changed, the **device will now respond using the new ID.**
It is recommended that each device in a system has a **unique ID** to prevent conflicts.

7.6.4 Protocol Change Command

This command is used to change the communication protocol of the device.
In RFIDAX devices, this command allows switching between predefined communication protocols.
The device saves the newly selected protocol in **Flash Memory** and uses this protocol for all future communications.
If the device receives an unsupported protocol value, it returns an error.
The selected protocol will be used for **all device communications**.

Sent Message Format (Protocol Change Command)

Sent Message Format (Protocol Change Command)						
HEADER	DEVICE_ ADDRESS	COMMAND_ TYPE	SUB_COMMAND _TYPE	NEW_PROTOCOL	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xAA	0x01	0x0E	0x02	0xXX	0xXX	0xXX

Field Descriptions

HEADER (0xAA) → Indicates the start of the command. This value is fixed.
DEVICE_ADDRESS (0x01) → The current address of the device to which the command is sent.
COMMAND_TYPE (0x0E) → Specifies device configuration operations.
SUB_COMMAND_TYPE (0x02) → Indicates the protocol change operation.
NEW_PROTOCOL (0xXX) → The new protocol value to be assigned. (See the supported CRC protocols below.)
CRC_HIGH & CRC_LOW (Optional) → Used for ensuring message integrity.

Supported Protocols

Protocol Code	Protocol Name
0x00	USB Protocol
0x01	PROFIBUS Protocol
0x02	MODBUS Protocol
0x03	KERMIT CRC16 Protocol
0x04	CCITT CRC16 Protocol
0x05	ISO/IEC 14443-A Protocol

Table 40 Supported Protocols

For a detailed explanation of CRC protocols, refer to "6.2.2 Data Integrity in Active Mode."

Important Note

If the device receives a protocol code not listed in the table above, it may return an "ERR_INVALID_PROTOCOL" error.

For error handling, refer to "6.4.5 Advanced Errors and System Management."

To maintain a stable connection, the new protocol must be correctly configured.

Response Message Format (Protocol Change Response)

Response Message Format (Protocol Change Response)					
HEADER	DEVICE_ADDRESS	STATUS_CODE_HIGH	STATUS_CODE_LOW	CRC_HIGH (Optional)	CRC_LOW (Optional)
0xBB	0x01	0x00	0x00	0xXX	0xXX

Field Descriptions (Response Message)

- **HEADER (0xBB)** → Indicates the beginning of the response message. It is a fixed value.
- **DEVICE_ADDRESS (0x01)** → Represents the address of the device sending the response.
- **STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00)** →
  - 0x00 0x00 → Indicates that the protocol has been successfully changed.
  - Any other error code → Indicates that the protocol change was unsuccessful. (For more details, refer to section *6.4.1 General System Status and Errors*.)
- **CRC_HIGH & CRC_LOW (Optional)** → Error-checking mechanism to ensure message integrity.

Steps of This Process

1. The user sends a command to change the device's communication protocol.
2. The device checks whether the specified protocol is valid.
3. If the protocol is valid, it is stored in Flash Memory, and the device starts using the new protocol.
4. Upon successful execution, the device responds with a status code of "0x00 0x00".
5. The response message includes CRC values to ensure message integrity.
6. If an invalid protocol value is sent, the device returns an error message.

Important Notes

- The new protocol will be used for all device communications.
- The device will continue using the newly saved protocol even after a restart.
- If an invalid or unsupported protocol value is sent, the device may return an "**ERR_INVALID_PROTOCOL**" error. (For more details, refer to section *6.4.1 General System Status and Errors*.)
- After changing the protocol, communication with the device must be continued using the new protocol.
- This command is designed to enhance the device’s flexibility in systems utilizing different communication methods.

1st Example: Protocol Update Process

Sent Message Format

Command Sent (Command to Update Protocol in RFID System)						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	NEW PROTOCOL	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0E	0x02	0x02	0x56	0x53

Field Descriptions

HEADER (0xAA): Indicates the beginning of the command. It is a fixed value.
DEVICE_ADDRESS (0x01): Represents the address of the device.
COMMAND_TYPE (0x0E): Main command type indicating device configuration operations.
SUB_COMMAND_TYPE (0x02): Specifies the protocol change operation.
NEW_PROTOCOL (0x02): The selected new protocol (In this example, the device switches to the MODBUS CRC16 protocol).
CRC_HIGH (0x56) & CRC_LOW (0x53): Error-checking mechanism used to ensure data integrity.

Received Message Format

Response (Protocol Update Response from RFID System)					
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH (Optional)	CRC LOW (Optional)
0xBB	0x01	0x00	0x00	0x00	0x75

Field Descriptions

HEADER (0xBB): Indicates the beginning of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01): Represents the address of the device sending the response.
STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00):
0x00 0x00 → Indicates that the protocol has been successfully changed.
If an error occurs, different error codes may be returned.
CRC_HIGH (0x00) & CRC_LOW (0x75): Error-checking mechanism used to ensure data integrity.

7.6.6 Baud Rate Update Process

Baud rate is a critical parameter that determines the communication speed of the device.
This command is used to change the device's current baud rate value.
Baud rate modification is performed to optimize communication performance, ensure compatibility with different systems, or set a specific data transmission speed.
The device must be restarted after the baud rate update.

7.6.6.1 Supported Baud Rate Values

Below is a list of baud rate speeds supported by the device and their corresponding index values:

Baud Rate (bps)	Baud Rate Index (0xXX)
600	0x00
1200	0x01
2400	0x02
4800	0x03
9600	0x04
14400	0x05
19200	0x06
28800	0x07
38400	0x08
56000	0x09
57600	0x0A
115200	0x0B
128000	0x0C
256000	0x0D

Table 41 Supported Baud Rate Values

Baud rate selection is made using the corresponding index value.
The user must insert the index value corresponding to the desired baud rate into the command.

7.6.6.2 Sent Message Format

Command Sent (Command to Update Baud Rate in RFID System)

Command Sent (Command to Update Baud Rate in RFID System)						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	NEW BAUD RATE INDEX	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0E	0x03	0xXX	0xXX	0xXX

Field Descriptions

- **HEADER (0xAA):** Indicates the beginning of the command. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** Represents the address of the device receiving the command.
- **COMMAND_TYPE (0x0E):** Main command type indicating device configuration operations.
- **SUB_COMMAND_TYPE (0x03):** Specifies the baud rate change operation.
- **NEW_BAUD_RATE_INDEX (0xXX):** The newly selected baud rate index value.
  (One of the indices provided in the table above must be used.)
- **CRC_HIGH (0xXX) & CRC_LOW (0xXX):** Error-checking mechanism used to ensure data integrity.

Response Message Format

Response (Baud Rate Update Response from RFID System)

Response (Baud Rate Update Response from RFID System)					
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH (Optional)	CRC LOW (Optional)
0xBB	0x01	0x00	0x00	0xXX	0xXX

Response Message Descriptions

- **HEADER (0xBB):** Indicates the beginning of the response message. It is a fixed value.
- **DEVICE_ADDRESS (0x01):** Represents the address of the device sending the response.
- **STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00):**
  - **0x00 0x00** → Baud rate successfully changed.
  - If an error occurs, different error codes may be returned.
- **CRC_HIGH (0xXX) & CRC_LOW (0xXX):** Error-checking mechanism used to ensure data integrity.

Steps in This Process

1. The user sends the command containing the new baud rate value to the device.
2. The device checks whether the selected baud rate is valid.
3. Once the baud rate update is successfully completed, the device returns a response message.
4. The new baud rate is stored in the device's memory, and the communication speed is changed.
5. **The device must be restarted after the baud rate update.**

Important Notes

- **The device must be restarted after the baud rate update.**
- If an unsupported baud rate is selected, the device may return the "**ERR_INVALID_BAUD_RATE**" error.
  - For details, refer to **section** 6.4.1 General System Status and Errors.
- **Baud rate changes can immediately affect communication!**
  - If the new baud rate is not supported by the system, communication may be lost.
- If communication with the device is lost, **previous baud rate values should be tested** to restore connection.

1st Example: Baud Rate Update Process

This example demonstrates the structure and process of the command used to change the baud rate of the RFID device.
Baud rate is a **critical parameter** that determines the device's communication speed, and its proper configuration is essential for stable data transmission.

Sent Message Format

Command Sent (Command to Update Baud Rate in RFID System)						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	NEW BAUD RATE INDEX	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0E	0x03	0x0B	0xF4	0x4B

Sent Message Descriptions

HEADER (0xAA): Indicates the beginning of the command. It is a fixed value.
DEVICE_ADDRESS (0x01): The address of the device receiving the command.
COMMAND_TYPE (0x0E): The main command defining device configuration operations.
SUB_COMMAND_TYPE (0x03): Indicates the baud rate change operation.
NEW_BAUD_RATE_INDEX (0x0B): The selected new baud rate index (**115200 baud**).
CRC_HIGH (0xF4) & CRC_LOW (0x4B): Error-checking mechanism ensuring data integrity

Received Message Format

Response (Baud Rate Update Response from RFID System)					
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH (Optional)	CRC LOW (Optional)
0xBB	0x01	0x00	0x00	0x5C	0x3E

Response Message Descriptions

HEADER (0xBB): Indicates the beginning of the response message. It is a fixed value.
DEVICE_ADDRESS (0x01): The address of the responding device.
STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00):
0x00 0x00 → Baud rate successfully changed.
If an error occurs, different error codes may be returned.
CRC_HIGH (0x5C) & CRC_LOW (0x3E): Error-checking mechanism ensuring data integrity.

Steps in This Process

1-The user sends the command containing the new baud rate value to the device.
2-The device verifies whether the new baud rate value is valid.
3-After the baud rate update is successfully completed, the device returns a response message.
4-After restarting the device, the new baud rate is stored in the device's memory, and all further communications occur at this speed.

Important Notes:

When assigning a new baud rate, the configuration should be adjusted accordingly to maintain communication.
If an incorrect baud rate is assigned, communication issues may arise.
The baud rate change takes effect after the device is restarted.

info@rfidax.com
www.rfidax.com

7.7 Factory Reset Commands

RFIDAX devices use **Factory Reset Commands** to reset all settings, erase user data, and format device memory.
These commands permanently restore the device to factory defaults and **erase all stored configurations irreversibly**.
The factory reset commands perform the following operations:

- Reset **all device settings**.
- **Read or modify** the Format ID stored in flash memory.

Factory Reset Command List

Command Code	Sub Command Code	Operation
0x0F	0x01	Reset all settings to factory defaults
0x0F	0x02	Read Format RFID Card ID
0x0F	0x03	Set/Reset Format Flag
0x0F	0x04	Manually Write Format ID to Flash Memory

Table 42 Factory Reset Command List

Important Notes

These operations are irreversible, and all configurations will be lost after the factory reset.
Before performing this operation, it is recommended to back up existing data.

info@rfidax.com
www.rfidax.com

7.7.1 Reset All Settings to Factory Defaults Command

This command resets all configuration settings on the RFIDAX device to factory defaults.
The default **Device ID, Baud Rate, and Protocol settings** are restored and written to Flash Memory.
After the process is completed, the **device automatically restarts** and continues operation with factory settings.

Usage Scenarios:

Reset all **device configurations** to factory defaults.
Revert misconfigured settings to original values.
Reset **communication settings** to standard factory configuration.
Fix system issues by restoring factory settings.

Command Sent (Reset Device to Factory Defaults)

Command Sent (Command to Reset Device to Factory Defaults)					
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0F	0x01	0xXX	0xXX

Field Descriptions:

- **HEADER (0xAA)** → Marks the beginning of the command. Constant value.
- **DEVICE_ADDRESS (0x01)** → Address of the device receiving the command.
- **COMMAND_TYPE (0x0F)** → Indicates the main **Factory Reset** command category.
- **SUB_COMMAND_TYPE (0x01)** → Specifies the **reset to factory defaults** operation.
- **CRC_HIGH & CRC_LOW** → Used for message integrity verification.

Response Message Format (Factory Reset Confirmation)

Response Message Format (Factory Reset Confirmation)					
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH (Optional)	CRCRC LOW (Optional)
0xBB	0x01	0x00	0x00	0xXX	0xXX

Field Descriptions (Response Message):

- **HEADER (0xBB)** → Marks the beginning of the response message. Constant value.
- **DEVICE_ADDRESS (0x01)** → Address of the responding device.
- **STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00)**
  - **0x00 0x00** → Factory reset successfully completed.
- **CRC_HIGH & CRC_LOW** → Used for message integrity verification.

Process Steps:

1-**User sends the factory reset command to the device.**

2-**The device restores factory settings in Flash Memory:**

- **Device ID =** 0x01
- **Baud Rate =** 9600 (**Baud Rate Index:** 0x04)
- **Protocol =** KERMIT CRC16 (**0x03**)

3-**A "SUCCESSFUL" message is returned upon successful factory reset.**

4-**The device automatically restarts** and continues operation with **default settings**.

Important Notes:

This operation resets all device settings and is irreversible!
Since communication settings are reset, ensure the new settings are configured properly.
⚠After a reset, the device will communicate at 9600 Baud Rate and KERMIT CRC16 protocol.
If an invalid command is sent, the device will return the error: "**ERR_FLASH_WRITE_FAILURE**".

1. Example: Factory Reset Operation

This example demonstrates the command sent to reset all device settings to factory defaults and the response received from the system.

Command Sent (Command to Reset Device to Factory Defaults)

Command Sent (Command to Reset Device to Factory Defaults)					
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	CRC HIGH	CRC LOW
0xAA	0x01	0x0F	0x01	0x31	0x32

Field Descriptions:

- HEADER (0xAA) → Marks the beginning of the command.
- DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
- COMMAND_TYPE (0x0F) → Indicates the factory reset command category.
- SUB_COMMAND_TYPE (0x01) → Specifies the reset to factory defaults operation.
- CRC_HIGH (0x31) & CRC_LOW (0x32) → Used for message integrity verification.

Response (Factory Reset Confirmation from RFID System)

Response (Factory Reset Confirmation from RFID System)					
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH	CRC LOW
0xBB	0x01	0x00	0x00	0x47	0x89

Field Descriptions (Response Message):

- HEADER (0xBB) → Marks the beginning of the response message.
- DEVICE_ADDRESS (0x01) → Address of the responding device.
- STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00)
  - 0x00 0x00 → Factory reset successfully completed.
- CRC_HIGH (0x47) & CRC_LOW (0x89) → Used for message integrity verification.

7.7.2 Format RFID Card ID Read Command

This command is used to read the format ID information of an RFID card.
The device retrieves the stored format ID value from memory and returns it as a response.
Usage Purposes
This command is used for the following purposes:

- Reading the format ID value of RFID cards
- Verifying the existing format ID information
- Checking the validity of format operations

Command Sent (Command to Read Format RFID Card ID in RFID System)

Command Sent (Command to Read Format RFID Card ID in RFID System)					
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0F	0x02	0xXX	0xXX

Field Descriptions:

- HEADER (0xAA) → Marks the beginning of the command.
- DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
- COMMAND_TYPE (0x0F) → Indicates the factory reset command category.
- SUB_COMMAND_TYPE (0x02) → Specifies the format RFID ID read operation.
- CRC_HIGH (XX) & CRC_LOW (XX) → Used for message integrity verification.

Response (Format RFID Card ID Read Response from RFID System)

Response (Format RFID Card ID Read Response from RFID System)					
BAŞLIK	CİHAZ ADRESİ	MESAJ TÜRÜ	VERİ (RFID Kimliği Biçimi)	CRC YÜKSEK	CRC DÜŞÜK
0xAA	0x01	0x60	0xXX 0xXX 0xXX 0xXX	0xXX	0xXX

Field Descriptions (Response Message):

- HEADER (0xAA) → Marks the beginning of the response message.
- DEVICE_ADDRESS (0x01) → Address of the responding device.
- MESSAGE_TYPE (0x60) → Message code for format RFID ID read operation.
  - For details on the message type, please refer to "6.7.2.1.5 Format ID Read Message and Message Type Information."
- DATA (0xXX 0xXX 0xXX 0xXX) → Format RFID ID value.
- CRC_HIGH (0xXX) & CRC_LOW (0xXX) → Used for message integrity verification

Steps in This Process

1-The user sends a command to read the format RFID ID.

2-The device retrieves the stored format ID value from Flash Memory.

3-The retrieved ID information is returned in the response message.

4-CRC values are added to ensure message integrity.

1.Example: Reading Format RFID Card ID

This example demonstrates the command used to read the format ID of an RFID card and the response received from the device.

Sent Message (Command to Read Format RFID ID in RFID System)

Sent Message (Command to Read Format RFID ID in RFID System)					
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	CRC HIGH	CRC LOW
0xAA	0x01	0x0F	0x02	0x01	0x51

Field Descriptions:

- HEADER (0xAA) → Marks the beginning of the command.
- DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
- COMMAND_TYPE (0x0F) → Indicates the factory reset command category.
- SUB_COMMAND_TYPE (0x02) → Specifies the format RFID ID read operation.
- CRC_HIGH (0x01) & CRC_LOW (0x51) → Used for message integrity verification.

Response (Format RFID ID Read Response from RFID System)

Response (Format RFID ID Read Response from RFID System)					
HEADER	DEVICE ADDRESS	MESSAGE TYPE	DATA (Format RFID ID)	CRC HIGH	CRC LOW
0xAA	0x01	0x60	0x93 0x32 0xEF 0xF6 0xE2 0x20	0xE2	0x20

Field Descriptions (Response Message):

- HEADER (0xAA) → Marks the beginning of the response message.
- DEVICE_ADDRESS (0x01) → Address of the responding device.
- MESSAGE_TYPE (0x60) → Message code for format RFID ID read operation.
- DATA (0x93 0x32 0xEF 0xF6 → Format RFID ID value.
- CRC_HIGH (0xE2) & CRC_LOW (0x20) → Used for message integrity verification.

7.7.3 Format Flag Set/Reset Command

This command is used to set (active) or reset (passive) the format flag.
The format flag is a control mechanism that determines whether the format process of the device is active.

Usage Purposes

Used to determine whether there is a Format ID card with a flag.
Controls how the system behaves based on the value of the format flag.
When this flag is passive, the device requests a Format TAG from the user.
When this flag is active, the device continues normal operation.

Sent Message Format (Format Flag Set/Reset Command)

Message Sent (Command to Set/Reset Format Flag in RFID System)						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	Format Flag	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0F	0x03	0xXX	0xXX	0xXX

Field Descriptions

HEADER (0xAA) → Marks the beginning of the command. This is a fixed value.
DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
COMMAND_TYPE (0x0F) → The main command that defines factory reset operations.
SUB_COMMAND_TYPE (0x03) → Specifies the format flag set/reset operation.
FORMAT_FLAG (0xXX) → Determines the flag value:
0x00 → Resets the format flag (clears the flash memory, cancels the format process).
0x01 → Sets the format flag (writes 0x00000001 to flash memory, activates the format process).
CRC_HIGH & CRC_LOW (Optional) → Used for message integrity verification.

Device Response Behavior

This process does not return a response message; instead, the device exhibits specific behaviors based on the flag status.

Process Steps

Set Operation (Activating - When 0x01 is Sent)

1. The user sends the command to activate the format process.
2. The device updates the related format flag in flash memory to **0x00000001**.
3. The format flag is set, and the device **does not request a format tag**.
4. The Format Tag ID is set to **FF FF FF FF**.
5. The device continues normal operation.

Reset Operation (Deactivating - When 0x00 is Sent)

1. The user sends the command to deactivate the format tag flag.
2. The device updates the format flag stored in flash memory to **0xFFFFFFFF**.
3. The device starts requesting a **format tag**.
4. Until the format tag is read, the device continuously requests it.
   If the format tag is not read, the device **returns the error "ERR_RESET_CARD_NOT_READ"**.
   (Please refer to "6.4.5 Advanced Errors and System Management" for details.)
   The only way to resolve this error is either by defining a **format tag** or setting the **format flag back to 0x01**.

7.7.3.1 Format Flag Reset (0x00 - Deactivation)

This section presents the command sent to reset the device's format flag and the response received from the device.

Sent Message (Command to Reset Format Flag in RFID System)

Command Sent (Command to Reset Format Flag in RFID System)						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	Format Flag	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0F	0x03	0x00	0x72	0x10

Field Descriptions

HEADER (0xAA) → Marks the beginning of the command. This is a fixed value.
DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
COMMAND_TYPE (0x0F) → The main command that defines factory reset operations.
SUB_COMMAND_TYPE (0x03) → Specifies the format flag set/reset operation.
FORMAT_FLAG (0x00) → Resets the format flag (clears the flash memory, cancels the format process).
CRC_HIGH (0x72) & CRC_LOW (0x10) → Used for message integrity verification.

Incoming Response Message (Response Containing Error Code)

Incoming Response Message (Response Containing Error Code)						
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH	CRC LOW	ERROR CODE DESCRIPTION
0xBB	0x01	0x00	0x71	0x32	0x88	ERR_RESET_CARD_NOT_READ -

Field Descriptions

HEADER (0xBB) → Marks the beginning of the response message. This is a fixed value.
DEVICE_ADDRESS (0x01) → The address of the device sending the response.
STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x71) →
0x00 0x71 indicates that the device has reset the format flag but failed to read the format tag (ERR_RESET_CARD_NOT_READ).
CRC_HIGH (0x32) & CRC_LOW (0x88) → Used for message integrity verification.

Error Code Description (ERR_RESET_CARD_NOT_READ)

This error means that after resetting the format flag, the device **did not read a format tag**.
The device **cannot continue normal operation without a format tag**.
The system will continuously search for a format tag and will keep returning **"ERR_RESET_CARD_NOT_READ"**.
(Refer to "6.4.5 Advanced Errors and System Management" for details.)

Solution Methods

To resolve this error, one of the following two solutions must be applied:

1-Read a Format Tag

The format tag expected by the device must be scanned and registered using an RFID reader.
Once the correct format tag is scanned, the device will return to normal operation.

2-Set the Format Flag Again

The format flag should be set back to **"0x01"** to exit format mode.
To achieve this, the **Format Flag Set (0x01) command** must be sent to the device.

7.7.3.2 Setting the Format Flag (0x01 - Activation)

This example presents the command sent to activate (set) the device's format flag and the device's response behavior.

Sent Message (Command to Set Format Flag in RFID System)

Command Sent (Command to Set Format Flag in RFID System)

Command Sent (Command to Set Format Flag in RFID System)						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	FORMAT FLAG	CRC HIGH	CRC LOW
0xAA	0x01	0x0F	0x03	0x01	0x62	0x31

Field Descriptions

- HEADER (0xAA) → Marks the beginning of the command. This is a fixed value.
- DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
- COMMAND_TYPE (0x0F) → The main command that defines factory reset operations.
- SUB_COMMAND_TYPE (0x03) → Specifies the format flag set/reset operation.
- FORMAT_FLAG (0x01) → Activates the format flag (writes 0x00000001 to Flash memory).
- CRC_HIGH (0x62) & CRC_LOW (0x31) → Used for message integrity verification.

Device Response

- The device **does not** return a response message to this command.
- The device updates the format flag in Flash memory to **"0x00000001"**.
- The format flag is set, and the device **does not require a format tag**.
- The format tag ID is automatically set to **"FF FF FF FF"**.
- The device **continues normal operation** without any format requests.

Steps in This Process

Setting the Format Flag (0x01 Sent - Activation)

1-The user sends the command to activate the format flag.

2-The device updates the format flag in Flash memory to **0x00000001**.

3-The format flag is set, and the device **stops requiring a format tag**.

4-The format tag ID automatically becomes **FF FF FF FF**.

5-The device continues **normal operation**.

Important Notes

- This command changes the **format flag status** but does **not** fully format the device.
- If the format flag is reset (0x00), the device will continuously return errors until a format tag is read.
- When the format flag is set (0x01), the device operates normally and **does not request a format tag**.
- If the format flag is mistakenly reset, the device can only resume normal operation by either **scanning a format tag** or **sending the Format Flag Set (0x01) command again**.

Comparison of Format Flag States

Operation	Result	Error Code
Reset Format Flag (0x00)	Device starts waiting for a format tag and continuously returns errors.	ERR_RESET_CARD_NOT_READ (0x00 0x71)
Set Format Flag (0x01)	Device continues normal operation, does not request a format tag.	Device does not return a response.

Table 43 Format Flag Set (Active) or Reset (Passive) Status Table)

Conclusion

- Resetting the format flag **triggers an error message** and causes the device to **wait for a format tag**.
- To recover from this issue, **either scan a format tag** or **set the flag back to "0x01"** using the appropriate command.

7.7.4 Manually Writing Format ID to Flash Memory

This command is used to manually write a user-defined Format ID directly into the Flash memory.
This ID is used in the device's format mechanism and is stored in Flash memory for specific operations.

Usage Purposes

Writing a user-defined Format ID into the device's Flash memory manually.
Ensuring the device has a predefined authentication mechanism during formatting.
Verifying whether the specified Format ID is accepted by the system.

Sent Message Format

Sent Message Format						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	FORMAT ID (4 Bytes)	CRC HIGH (Optional)	CRC LOW (Optional)
0xAA	0x01	0x0F	0x04	0xXX 0xXX 0xXX 0xXX	0xXX	0xXX

Field Descriptions

HEADER (0xAA) → Marks the beginning of the command. This is a fixed value.
DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
COMMAND_TYPE (0x0F) → The main command defining factory reset operations.
SUB_COMMAND_TYPE (0x04) → Specifies the manual Format ID writing operation to Flash memory.
FORMAT_ID (0xXX 0xXX 0xXX 0xXX) → The 4-byte Format ID data to be stored in Flash memory.
CRC_HIGH & CRC_LOW (Optional) → Used for message integrity verification.

Incoming Message Format (Format ID Write Response)

Incoming Message Format (Format ID Write Response)					
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH (Optional)	CRC LOW (Optional)
0xBB	0x01	0x00	0x00	0xXX	0xXX

Field Descriptions

HEADER (0xBB) → Marks the beginning of the response message. This is a fixed value.
DEVICE_ADDRESS (0x01) → The address of the device sending the response.
STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00)

0x00 0x00 → The Format ID was successfully written to Flash memory.
Other error codes → The Format ID write operation failed. (Refer to "6.4.1 General System Status and Errors" for details.)

CRC_HIGH & CRC_LOW (Optional) → Used for message integrity verification.

Process Steps

1-The user sends the command to store a specified Format ID into Flash memory.
2-The device verifies whether the incoming Format ID is valid.
3- If the Format ID is in a valid **HEX format**, the device erases Flash memory and writes the new ID.
4-Upon successful completion, the device responds with the **"0x00 0x00"** status code.
5-The device will use this stored Format ID for future formatting operations.

Important Notes

The Format ID must be in HEX format only. Sending a value greater than **0xFF** may result in an ERR_INVALID_HEX_DATA error. (Refer to 6.4.5 Advanced Errors and System Management for details.)
Flash memory erasure is irreversible. Once a new Format ID is written, the previous value **cannot be restored**.
If an invalid Format ID is sent, the device will reject the operation and return an error code.
Once successfully written, the device will use this new Format ID in all future format operations.

1st Example: Manually Writing a Format ID to Flash Memory

This example demonstrates the command sent to manually store a user-defined Format ID into Flash memory and the corresponding response.

Sent Message (Format ID Write Command)

Sent Message (Command to Write Format ID to Flash Memory)						
HEADER	DEVICE ADDRESS	COMMAND TYPE	SUB COMMAND TYPE	FORMAT ID (4 Bytes)	CRC HIGH	CRC LOW
0xAA	0x01	0x0F	0x04	0x13 0x75 0x41 0xFC	0x71	0x22

Field Descriptions

- HEADER (0xAA) → Marks the beginning of the command. This is a fixed value.
- DEVICE_ADDRESS (0x01) → The address of the device receiving the command.
- COMMAND_TYPE (0x0F) → The main command defining factory reset operations.
- SUB_COMMAND_TYPE (0x04) → Specifies the manual Format ID writing operation to Flash memory.
- FORMAT_ID (0x13 0x75 0x41 0xFC) → The 4-byte Format ID data to be stored in Flash memory.
- CRC_HIGH (0x71) & CRC_LOW (0x22) → Used for message integrity verification.

Incoming Message (Format ID Write Response)

Response Message (Format ID Write Response)					
HEADER	DEVICE ADDRESS	STATUS CODE HIGH	STATUS CODE LOW	CRC HIGH	CRC LOW
0xBB	0x01	0x00	0x00	0xXX	0xXX

Field Descriptions

- HEADER (0xBB) → Marks the beginning of the response message. This is a fixed value.
- DEVICE_ADDRESS (0x01) → The address of the device sending the response.
- STATUS_CODE_HIGH (0x00) & STATUS_CODE_LOW (0x00)
  - 0x00 0x00 → The Format ID was successfully written to Flash memory.
  - If an error occurs, a different error code will be returned.
- CRC_HIGH (0xXX) & CRC_LOW (0xXX) → Used for message integrity verification.