SONY

FeliCa Lite-S

# FeliCa Lite-S

# User's Manual

Version 1.21

No. M741-E01-21

# Introduction

This document describes the protocol specifications and the command specifications of any contactless IC card that uses FeliCa Lite-S technology.

The purpose of this document is to provide basic information about the protocol specifications and the command specifications to customers who are engaged in the development of the Reader/Writer and application software that use FeliCa Lite-S technology.

The objects of the descriptions in this document are the FeliCa Lite-S-based contactless IC cards and IC chips sold by Sony.

For details of the FeliCa Standard series, FeliCa Lite series, and FeliCa Plug series, see the following website:

http://www.sony.net/Products/felica/business/tech-support/index.html

For details of the differences between FeliCa Lite-S documents and FeliCa Lite documents, see “Differences Between the FeliCa Lite Documents and FeliCa Lite-S Documents”.

If you have any questions about the development of application software that is compatible with mobile FeliCa cards, please contact FeliCa Networks, Inc. (info-fn@FeliCaNetworks.co.jp).

In this document, products of FeliCa Lite-S series are expressed as “FeliCa Lite-S”, and FeliCa Lite-S card is expressed as “card”.

The content of this document does not guarantee the correct operation of system with all existing or future FeliCa Lite-S products.

FeliCa technology refers to the following standards:

- JIS X 6319-4: Specification of implementation for integrated circuit(s) cards – Part 4: High speed proximity cards
- ISO/IEC 18092: Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol-1 (NFCIP-1)
- NFC Forum: http://www.nfc-forum.org/

# Contents

# 1 Overview

Chapter 1 (this chapter) outlines the organization of this document, as follows:
Chapter 2 describes the communication protocol of FeliCa Lite-S.
Chapter 3 describes the FeliCa Lite-S file system.
Chapter 4 describes the commands used by FeliCa Lite-S.
Chapter 5 describes the security functionality of FeliCa Lite-S.
Chapter 6 describes the inspection of FeliCa Lite-S.
Chapter 7 describes the issuance of FeliCa Lite-S.

## 1.1 Notational conventions

This section outlines the notation in this document.
The following rules are applied as the numerical notation:

| | |
|---|---|
| Binary values | "b" is appended to a binary value (e.g., 0101b). |
| Hexadecimal values | "h" is appended to a hexadecimal value (e.g., FFFFh). |
| Decimal values | Nothing is appended to a decimal value (e.g., 10). |
| Bit notation | Bits are denoted in sequence from most-significant-bit (MSB) on the left to least-significant-bit (LSB) on the right. |
| ALL_Xb | Denotes all bits (e.g., ALL_0b, where all bits are 0b). |
| ALL_XXh | Denotes all Bytes (e.g., ALL_FFh, where all Bytes are FFh). |
| Byte order | Big Endian, unless otherwise specified. |

In figures, Byte strings and bit strings are denoted as shown in Figure 1-1, Figure 1-2, and Figure 1-3.

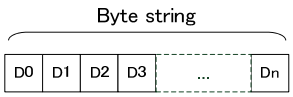



**Figure 1-1: Graphic notation of Byte string**

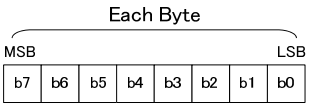



**Figure 1-2: Graphic notation of bit string**

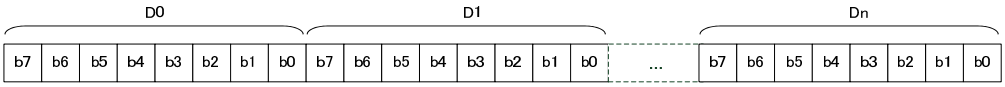



**Figure 1-3: Graphic notation of Byte string and bit string**

When referring to specific Bytes or bit in figures, the following notation is used:

| | |
|---|---|
| upper 2 Bytes | Indicates 2 Bytes, from D0 to D1, inclusive.<br>Unless otherwise specified, D0 is the most significant Byte. |
| D0-D15 | Indicates 16 Bytes from D0 to D15, inclusive. |
| upper 6 bits | Indicates 6 bits from b7 to b2, inclusive. |
| b5-b3 | Indicates 3 bits from b5 to b3, inclusive. |

# 2 Communication protocol

This chapter describes the communication protocol used for communication with FeliCa Lite-S and is organized as follows:

- Physical layer
  This layer defines the physical and electrical characteristics of data transfer.
- Data link layer
  This layer defines the data transfer method and the error detection scheme.
- Application layer
  This layer defines the specifications and functions of data strings to be handled as commands.

## 2.1 Physical layer

Table 2-1 shows the transmission characteristics of the physical layer of RF communication with FeliCa Lite-S.

**Table 2-1: Transmission characteristics of physical layer of RF communication interface**

| Item | Description |
|---|---|
| Data transfer method | Half duplex, synchronous system |
| Carrier frequency (fc) | 13.56 MHz |
| Modulation method | ASK |
| Bit coding | Manchester code, MSB first |
| Data transfer rate | fc/64 (approx. 212 kbps): herein after “212kbps”<br>fc/32 (approx. 424 kbps): herein after “424kbps”<br>NOTE When a command is received at 212 kbps/424kbps, a response is returned at 212kbps/424kbps, respectively. |

# 2.2 Data link layer

Data transfer between the Reader/Writer and the card is performed on a packet-by-packet basis, as defined in the data link layer. For definitions of fields in a packet and the packet structure, see Table 2-2 and Figure 2-1.

**Table 2-2: Definition of fields in a packet**

| Field name | Byte length | Definition |
|---|---|---|
| Preamble | 6 | (00 00 00 00 00 00)h |
| Sync code | 2 | (B2 4D)h |
| Data length (LEN) | 1 | Value of n (Byte length of Packet Data) + 1 (Byte length of LEN) |
| Packet Data | n | Command Packet Data / Response Packet Data<br>(to be defined on a command-by-command basis) |
| CRC | 2 | Checksum of data length and Packet Data based on CRC-CCITT (Big Endian)<br>Initial value: 00 00h<br>Generator polynomial: $X^{16} + X^{12} + X^5 + 1$ |

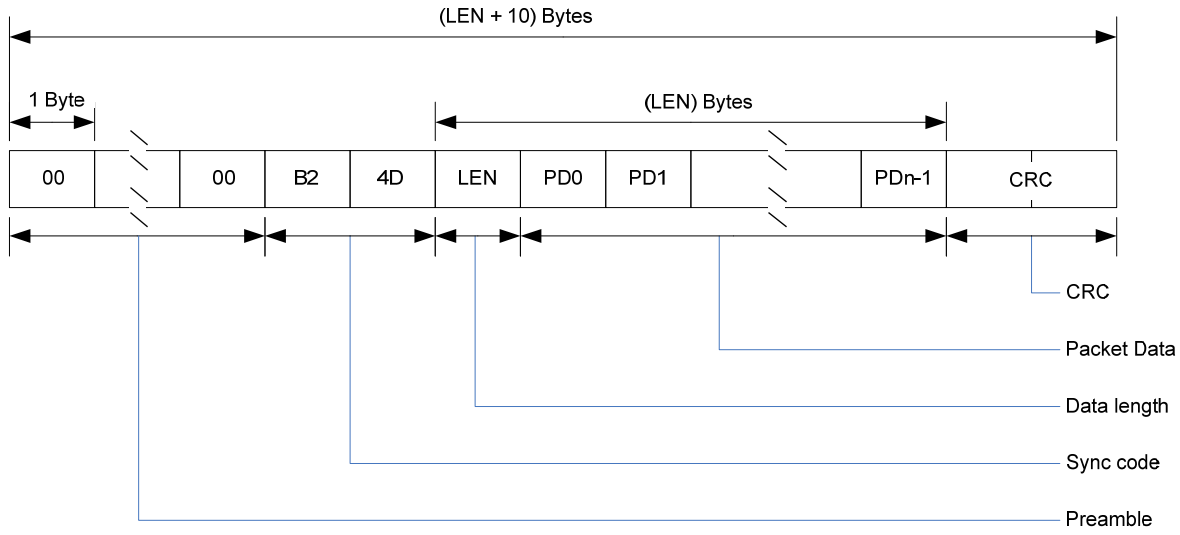



**Figure 2-1: Packet structure**

# 2.3 Application layer

This section describes the rules applied to Packet Data (i.e., the data contained in a packet). It also describes the rules that govern how the parameters contained in Packet Data are processed in accordance with the communication protocol.

In this document, Packet Data received by the card is known as command packet, and Packet Data transmitted from the card is known as response packet.

## 2.3.1 Command packet

A command packet consists of Command Code (i.e., the first Byte) followed by command data.

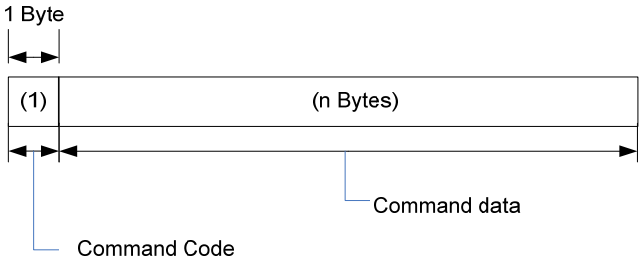



**Figure 2-2: Command packet**

**<Command Code>**

Command Code identifies the type of command. Table 2-3 shows the list of commands available. For detailed information about each command, see section 4.4 “Command specifications”.

**<Command data>**

Command data is defined on a command-by-command basis. For information about the contents to be defined, see section 4.4 “Command specifications”.

## 2.3.2 Response packet

A response packet consists of Response Code (i.e., the first Byte) and response data.

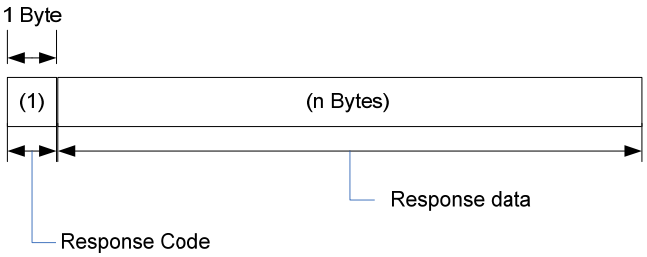



**Figure 2-3: Response packet**

**<Response Code>**

Response Code identifies the type of response. Table 2-3 shows the list of Response Code corresponding to the commands available.

**<Response data>**

Response data is defined on a command-by-command basis. For information about the contents to be defined, see section 4.4 “Command specifications”.

**Table 2-3: List of commands**

| Command name | Command code | Response code | Function overview |
|---|---|---|---|
| Polling | 00h | 01h | Use this command to acquire and identify a card. |
| Read Without Encryption | 06h | 07h | Use this command to read Block Data from authentication-not-required Service. |
| Write Without Encryption | 08h | 09h | Use this command to write Block Data to authentication-not-required Service. |

Commands other than these three commands and the Authentication1 command are not valid. If the FeliCa Lite-S receives a non-valid command, it will not respond.

For compatibility with FeliCa Standard, after receiving the Authentication1 command, FeliCa Lite-S ignores the Polling command, the Read Without Encryption command, and the Write Without Encryption command until the electrical power is shut off. FeliCa Lite-S will not respond even if it receives such commands.

### Manufacture ID and Manufacture Parameter

This section describes Manufacture ID (IDm) and Manufacture Parameter (PMm). IDm and PMm can be acquired as the response data to the Polling command. Figure 2-4 shows the configuration of IDm and PMm.

IDm and PMm cannot be changed from the value set when the IC is manufactured.

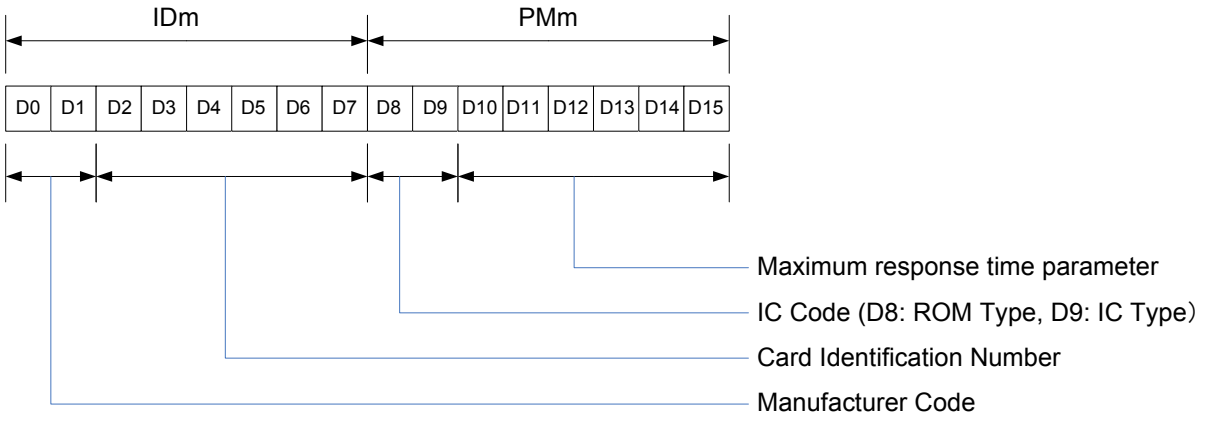



**Figure 2-4: IDm and PMm**

#### <Manufacture ID (IDm)>

Using Manufacture ID (IDm), the Reader/Writer identifies a card to be the counterpart of communication.

As shown in Figure 2-4, IDm consists of Manufacturer Code and Card Identification Number.

Manufacturer Code is the value that identifies the manufacturer of card or the provider of Manufacture ID (IDm). Therefore, do not use Manufacturer Code to identify FeliCa Lite-S.

#### <Manufacturer Parameter (PMm)>

As shown in Figure 2-4, PMm consists of the information (2 Bytes) known as IC Code (this code is used to identify the product) and the maximum response time parameter (6 Bytes; this parameter is used to determine the timeout period of each command). For a detailed description of each Byte from D10 to D15, see “Maximum response time”, on the next page.

IC Code consists of ROM Type and IC Type.

### Maximum response time

Timeout time is determined based on the period of time necessary for processing commands. Therefore (and because this period of time depends on the status of the card as well as on the type and content of each command), the Reader/Writer must dynamically determine the timeout time. In FeliCa technology, the maximum response time is determined by using the lower 6 Bytes of the PMm parameter. The card provides this parameter to the Reader/Writer, enabling the Reader/Writer to dynamically determine the timeout time. In FeliCa Lite-S, only D13 and D14 of the lower 6 Bytes of PMm are used. Fixed values are stored in D10, D11, D12, and D15. Be aware that the maximum response time differs per IC product.

The maximum response time parameter is the data string of 6 Bytes configured as shown in Figure 2-5. The Reader/Writer refers to 1 Byte of data located in PMm that corresponds with the command, and determines the maximum response time using the calculation formula shown in Figure 2-6. Additionally, acquisition of PMm is made possible by the Polling command.

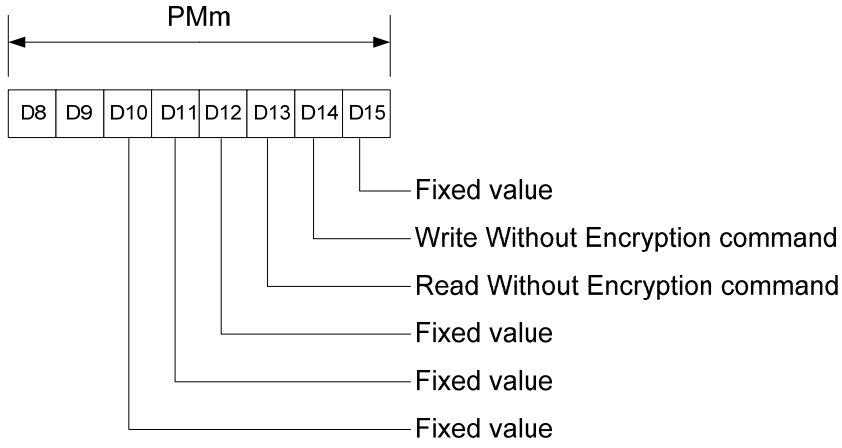
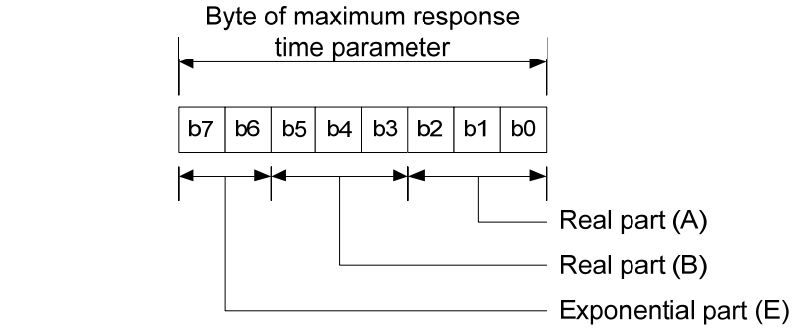



**Figure 2-5: Maximum response time parameter**




Maximum response time [ ms ] = $T \times [ ( B + 1 ) \times n + ( A + 1 ) ] \times 4^{E}$

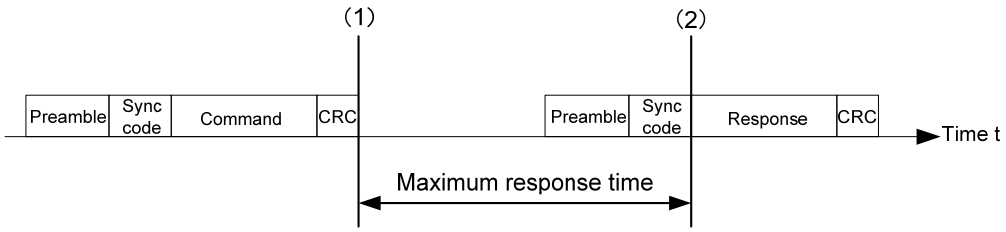
T = 256 × 16 / fc (about 0.3020ms)

n: Number of Block accessed by a command

**Figure 2-6: Calculation formula for determination of maximum response time**

In FeliCa Lite-S, the process time of each command is measured based on the definition of intervals shown in Figure 2-7, and the value of the maximum response time is determined.

For the Polling command, a response time different from the ones for the other commands is defined. For details, see section 2.3.3“Anti-collision process”.




(1) The point in time when transmission of all data in the command packet from the Reader/Writer is completed

(2) The point in time when transmission of the Sync code in the response packet from the card is completed

**Figure 2-7: Definition of maximum response time**

## 2.3.3 Anti-collision process

To identify a card, the Reader/Writer must poll an unspecified number of cards, by using the Polling command. If multiple cards exist within the range where communication between the Reader/Writer and the cards is possible, and if these cards respond to the Polling command simultaneously, however, the Reader/Writer is unable to correctly receive the responses returned from the cards. Therefore, FeliCa technology adopts a method known as Time Slot method to reduce the probability of collision between the responses returned from multiple cards.

**<Time Slot method>**

The Time axis includes sections that are divided into regular intervals; each such section is known as “Time Slot”. Both the Reader/Writer and the card have the same number of (i.e., “n”) Time Slots, and these slots are mutually synchronized. When a Polling command is received, the card selects Time Slot in a random manner and then transmits a response to the Polling command only in the selected slot. When the Reader/Writer performs polling to cards under the previously-mentioned assumptions, it is expected that the cards return responses to the polling in a random manner in each Time Slot. This reduces the probability of collision between responses to a Polling command sent to multiple cards.

In FeliCa technology, the start time of the first Time Slot is known as “response time (A)”, and the width (i.e., duration) of Time Slot is known as “response time (B)”. These response times are defined as follows:

- Response time (A)  512×64/fc (about 2.417) [ ms ]
- Response time (B)  256×64/fc (about 1,208) [ ms ]

The number of Time Slots (i.e., “n”) to be shared between the Reader/Writer and the card is specified by the Polling command. For details, see section 4.4.2 “Polling”.

Figure 2-8 shows an example of response times of the cards to the Polling command where the number of Time Slots is 4, and there are two cards within communication range of the Reader/Writer. This diagram shows the case where Card 1 selected slot #1 and Card 2 selected slot #3 of four Time Slots specified by the Reader/Writer.

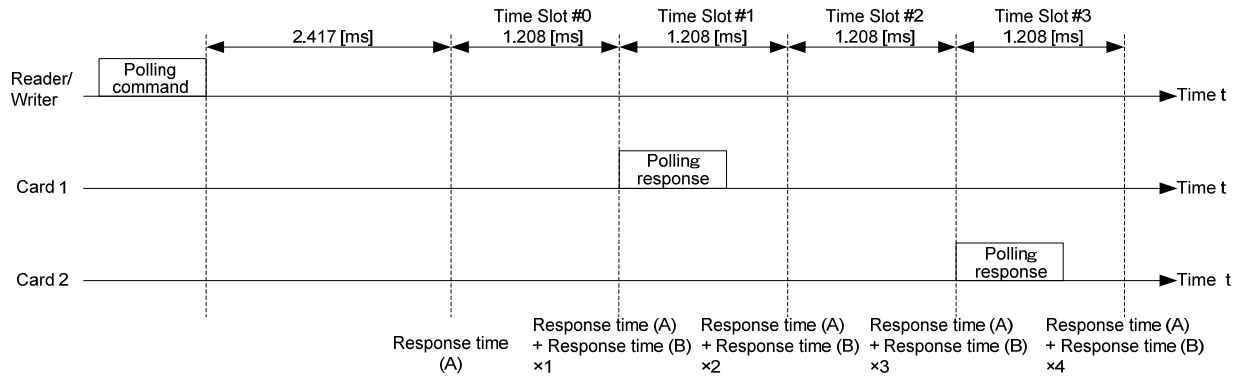



**Figure 2-8: Response time (where the number of Time Slots = 4)**

**<Identification of communication destination by IDm>**

When a response packet to the Polling command is received correctly, the Reader/Writer can acquire a parameter known as IDm, which is contained in the response packet to the Polling command. By setting IDm in the command packet, communication with a specific card becomes possible even if multiple cards are in proximity of the Reader/Writer. When a command packet is received, each card refers to IDm and, if such a card detects that the command packet is not addressed to itself, the card will not respond.

## 2.3.4 Polling Disable function

FeliCa Lite-S has the Polling Disable function. Using this function, you can prevent the card from responding to the Polling command. (If this setting is enabled, the card responds to the Read Without Encryption command and the Write Without Encryption command.) If you set an acquired card to Polling Disable status, you could issue the Polling command again to check the response. If there is no response returned, you can confirm that no card exists except for the acquired card.

If you reconfigure the card so as not to respond to the Polling command, or restart the card, it responds to the Polling command.

For details of the response setting to the Polling command, see section 3.1.15 “STATE”.

# 3 File system

A basic concept known as Service is introduced to the FeliCa Lite-S file system. Users can access Block using two types of Service: Service for Read/Write Access and Service for Read Only Access.

Note: Additionally, the concept of Area used in FeliCa Standard cards does not exist in FeliCa Lite-S.

This chapter describes the concept of the FeliCa Lite-S file system.

The concept of the file system is shown in Figure 3-1.

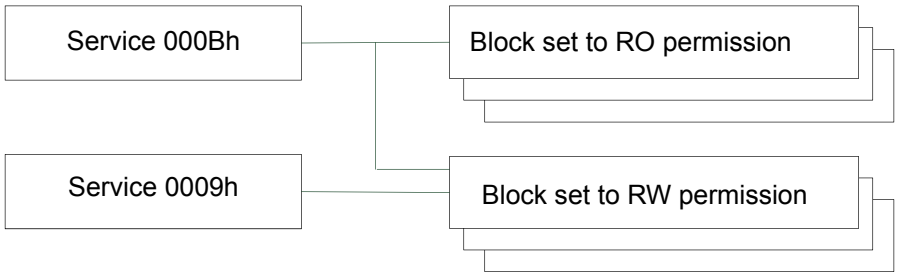



**Figure 3-1: Conceptual diagram of the FeliCa Lite-S file system**

## 3.1 Block

In this file system, management of non-volatile memory space is performed with the minimum recording unit of 16 Bytes. This minimum recording unit is known as Block.

All the user data is stored in Block. Access to the memory space from the user is performed on a Block-by-Block basis. Therefore, to store user data exceeding 16 Bytes, you must divide the data into more than one Block.

Access to Block can be achieved by using Service.

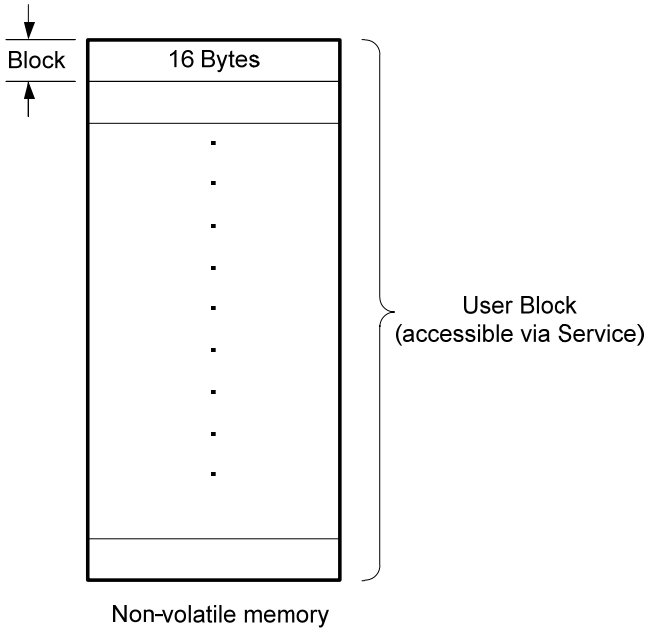



**Figure 3-2: Block in non-volatile memory**

## 3.1.1 List of Blocks

Table 3-1 shows the list of Blocks that can be used by users for FeliCa Lite-S.
Supported read methods and write methods are marked with ● for each Block.

**Table 3-1: List of Blocks**

| Block Number | Block name | Number of valid Bytes | Access permission for each issuance step | | | Method | | | | Note |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0th Issuance (at IC shipping) | 1st Issuance | 2nd Issuance | Read With MAC | Write With MAC | Read After Authentication | Write After Authentication | |
| 00h | S_PAD0 | 16 | RW | RW/RO[*1] | RW / RO[*2] | ● | ● | ● | ● | User Block |
| 01h | S_PAD1 | 16 | | | | ● | ● | ● | ● | |
| 02h | S_PAD2 | 16 | | | | ● | ● | ● | ● | |
| 03h | S_PAD3 | 16 | | | | ● | ● | ● | ● | |
| 04h | S_PAD4 | 16 | | | | ● | ● | ● | ● | |
| 05h | S_PAD5 | 16 | | | | ● | ● | ● | ● | |
| 06h | S_PAD6 | 16 | | | | ● | ● | ● | ● | |
| 07h | S_PAD7 | 16 | | | | ● | ● | ● | ● | |
| 08h | S_PAD8 | 16 | | | | ● | ● | ● | ● | |
| 09h | S_PAD9 | 16 | | | | ● | ● | ● | ● | |
| 0Ah | S_PAD10 | 16 | | | | ● | ● | ● | ● | |
| 0Bh | S_PAD11 | 16 | | | | ● | ● | ● | ● | |
| 0Ch | S_PAD12 | 16 | | | | ● | ● | ● | ● | |
| 0Dh | S_PAD13 | 16 | | | | ● | ● | ● | ● | |
| 0Eh | REG | 16 | | | | ● | ● | ● | ● | |
| 80h | RC | 16 | RW[*3] | RW[*4] | RW[*5] | ● | – | – | – | Authentication Function Block |
| 81h | MAC | 8 | RO | RO | RO | ● | – | – | – | |
| 82h | ID | Upper 8 | RO[*6] | RO | RO | ● | – | – | – | System Block |
| | | Lower 8 | RW | RO | RO | ● | – | – | – | |
| 83h | D_ID | 16 | RO | RO | RO | ● | – | – | – | |
| 84h | SER_C | 2 | RW[*7] | RO[*8] | RO[*9] | ● | – | – | – | |
| 85h | SYS_C | 2 | RO[*10] | RO[*11] | RO[*12] | ● | – | – | – | |
| 86h | CKV | 2 | RW[*13] | RW/RO[*14] [*15] | RW/RO[*16] [*17] | ● | ● | – | – | |
| 87h | CK | 16 | RW[*18] | RW/RO[*19] [*20] | RW/RO[*21] [*22] | ● | ● | – | – | |
| 88h | MC | 13 | RW | RW[*23] | RO | ● | – | – | – | |

| Block Number | Block name | Number of valid | Access permission for each issuance Method | | | | | | | Note |
|---|---|---|---|---|---|---|---|---|---|---|
| 90h | WCNT | 3 | RO | RO | RO | ● | – | – | – | Authentication Function Block |
| 91h | MAC_A | Read: 8<br>Write: 11 | RW | RW | RW | ● | – | – | – | |
| 92h | STATE | 2 | RW | RW | RW | ● | ● | – | – | |
| A0h | CRC_CHECK | 1 | RO | RO | RO | ● | – | – | – | Inspection Block |

RO=Read-Only

RW=Read and Write

*1, *2, *14, *16, *19, *21 Depends on the MC settings.

*3, *4, *5, *18, *20, *22 Values read from a card are ALL_00h.

*6 Data-write is possible. When data-read is performed immediately after data-write, the written value is read. However, the value is not saved in non-volatile memory.

*7, *8, *9, *10, *11, *12, *13, *15, *17 Data-read is possible if only Service Attribute of Service Code specified in the command is correct.

*23 Rewrite of data is possible only for MC [0], [1], [6], [7], [8], [9], [10], [11], [12].

A unique number (i.e., Block Number) is assigned to each Block. Use this number to specify which Block you want to access. For each Block, you can set the access permission to determine whether “Read-Only (RO)” or “Read-Write (RW)” of data is allowed. For Block for which RW permission was set (i.e., RW Block), it is possible to read and rewrite data. For Block for which RO permission was set (i.e., RO Block), however, it is possible only to read data. Block Number not listed in Table 3-1 is not valid. Access to non-valid Block Number is ignored, and an error is set in Status Flag of Response Packet Data.

## 3.1.2 S_PAD0-13

### Scratch Pad 0 - 13 Block (Block Number: 00h – 0Dh)

14 Blocks from S_PAD0 to S_PAD13 are areas of 16 Bytes each, and users are allowed to use these areas at their discretion. When the IC is shipped from the factory, ALL_00h is stored in these Blocks. Allocation of data is as shown in Figure 3-3: Scratch Pad Block

.

This Block can be set to accept Simple Read or Read After Authentication. (Read With MAC is always available regardless of the setting.) It can be set to accept Simple Write, Write After Authentication, Write With MAC, or Write With MAC after authentication. It can also be set to write protection (read-only).

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| SPAD0-13 | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| SPAD0-13 | | | | | | | |

**Figure 3-3: Scratch Pad Block**

**Byte15-0: S_PAD0-13**

Store any arbitrary data in these Bytes.

## 3.1.3 REG

### Subtraction Register Block (Block Number: 0Eh)

This Block of 16 Bytes can have data written to it, but only in a conditional manner. When the IC is shipped from the factory, ALL_FFh is stored. In accordance with the following conditions, values can be rewritten. Allocation of data is as shown in Figure 3-4: Subtraction Register Block

.

This Block can be set to accept Simple Read or Read After Authentication. (Read With MAC is always available regardless of the setting.) It can be set to accept Simple Write, Write After Authentication, Write With MAC, or Write With MAC after authentication. It can also be set to write protection (read-only).

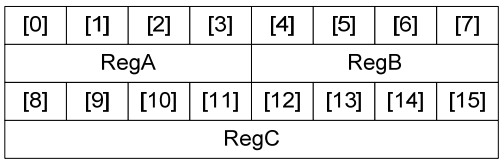



**Figure 3-4: Subtraction Register Block**

**Byte15-8: RegC**

Data-write is possible only when the conditions for data-write to RegA and RegB are satisfied. Any arbitrary data may be written.

**Byte7-4: RegB**

Data-write is possible only when data of a value is equal to or smaller than that stored in this register.

**Byte3-0: RegA**

Data-write is possible only when data of a value is equal to or smaller than that stored in this register.

Conditions for writing data to this Block are as follows:

Let the value stored in RegA be [A], and the value stored in RegB be [B]. Let the value to be written to RegA be [A'], and the value to be written to RegB be [B']. Only when A ≥ A' and B ≥ B', is rewrite of data to the whole Block performed. Consequently, rewriting of data to RegC area is possible only when these conditions are satisfied. Note that unsigned integer of 32 bits data is stored in RegA and RegB in Little Endian format, as shown in Figure 3-5: Endian of Subtraction Register Block

. RegA and RegB are handled as unsigned integers of 32 bits.

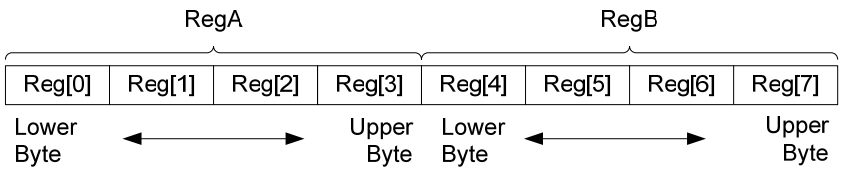



**Figure 3-5: Endian of Subtraction Register Block**

## 3.1.4 RC

### Random Challenge Block (Block Number: 80h)

This Block has a random number written to it, to be used for MAC generation functionality. Based on the value written to this Block, the session key necessary to generate the MAC is generated. Any arbitrary value may be written. When the data is read, in each case, ALL_00h is read. Data written to this Block is lost when the electric power is shut off. Immediately after switching the electric power on, an indefinite value is set in this Block. Therefore, make sure you write a specific value before the MAC becomes generated. Allocation of data is as shown in Figure 3-6.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| RC1 | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| RC2 | | | | | | | |

**Figure 3-6: Random Challenge Block**

**Byte15-8: RC2**

This is the plain text for session key SK2. Regardless of the actual values written, the values of the read data are always ALL_00h.

**Byte7-0: RC1**

This is the plain text for session key SK1, and it becomes the initial vector for generating the MAC. Regardless of the actual values written, the values of the read data are always ALL_00h.

## 3.1.5 MAC

### MAC Block (Block Number: 81h)

This Block stores the result of the MAC calculation to be read. This Block is read-only. If you write any data to this Block, it results in an error. MAC is generated for the simultaneously-read data. MAC is initialized at every data-read operation. Therefore, ALL_00h is read if only this Block is read. Allocation of data is as shown in Figure 3-7.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| MAC | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| - | | | | | | | |

**Figure 3-7: MAC Block**

**Byte15-8: Reserved**

In each case, 0 is read.

**Byte7-0: MAC**

Operation result of MAC is read.

## 3.1.6 ID

### ID Block (Block Number: 82h)

This Block stores ID. When the IC is shipped from the factory, the value of IDd is stored in the first 8 Bytes, and ALL_00h is stored in the second 8 Bytes.

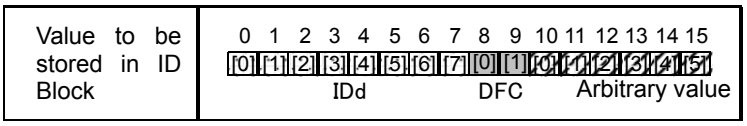
Allocation of data is as shown in Figure 3-8.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| IDd | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| ID | | | | | | | |

**Figure 3-8: ID Block**

**Byte15-8: ID**

Store the DFC in ID[8][9]. If the DFC is not used, store 0000h. Store an arbitrary value in the remaining 6 Bytes (see the following figure).




**Byte7-0: IDd**

The value of IDd is stored.

The value can be rewritten, so that inspection and issuance can be performed in the same way as for FeliCa Lite. The value, however, is not saved in non-volatile memory. Therefore, even if you rewrite the value, the initial value is restored when the electric power is shut off.

## 3.1.7 D_ID

### Device ID Block (Block Number: 83h)

This Block stores IDd and PMm. Values of IDd and IDm are the same.
Allocation of data is as shown in Figure 3-9.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| IDd | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| PMm | | | | | | | |

**Figure 3-9: Device ID Block**

**Byte15-8: PMm**

PMm is stored when the IC is shipped from the factory, and the value is unchangeable.

**Byte7-0: IDd**

IDd is stored when the IC is shipped from the factory, and the value is unchangeable.

## 3.1.8 SER_C

### Service Code Block (Block Number: 84h)

This Block stores Service Number (the most significant 10 bits of Service Code). When the IC is shipped from the factory, ALL_00h is stored. Use this Block without rewriting new data (i.e., keep this Block in its default status). Allocation of data is as shown in Figure 3-10.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| SER_C | | – | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| – | | | | | | | |

**Figure 3-10: Service Code Block**

**Byte15-2: Reserved**

In each case, 0 is read.

Writing of data is not valid.

**Byte1-0: Service Code (SER_C)**

Service Number is stored. The most significant 10 bits (SER_C[1] bit 7-0, SER_C[0] bit 7-6) are used to compare with the service code specified by the command packet.

NOTE The least significant 6 bits (SER_C[0] bit 5-0) are not used for the comparison.

When reading data from this Block, only Service Attribute of specified Service Code is checked. An arbitrary value can be set in Service Number.

## 3.1.9 SYS_C

### System Code Block (Block Number: 85h)

This Block stores System Code. When the IC is shipped from the factory, a predetermined value is stored. Rewrite of data is impossible. Allocation of data is as shown in Figure 3-11.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| SYS_C | | – | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| – | | | | | | | |

**Figure 3-11: System Code Block**

**Byte15-2: Reserved**

In each case, 0 is read.

Writing of data is not valid.

**Byte1-0: System Code (SYS_C)**

88h is stored in the first Byte, and B4h is stored in the second Byte. These are used for comparison with System Code specified by the Polling command.

FeliCa Lite-S returns a response to the Polling command in which the value (i.e., 88B4h) stored in this Block is specified as System Code. To store NDEF data (i.e., information that complies with the NDEF format specified by the NFC Forum) in User Blocks, however, the card shall respond to the Polling command in which 12FCh is specified as System Code. Therefore, a procedure is available to make FeliCa Lite-S return a response to the Polling command in which 12FCh is specified as System Code, regardless of the value stored in this Block. For details of this procedure, see section 3.1.12“MC”.

When reading data from this Block, only Service Attribute of specified Service Code is checked. An arbitrary value can be set in Service Number.

## 3.1.10 CKV

### Card Key Version Block (Block Number: 86h)

This Block stores Card Key Version. When the IC is shipped from the factory, ALL_00h is stored. Data can be rewritten only to the first 2 Bytes. Allocation of data is as shown in Figure 3-12.

Data-write after setting System Block to the rewrite prevention state can be set to write protection (read-only) or to accept Write With MAC.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| CKV | | – | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| – | | | | | | | |

**Figure 3-12: Card Key Version Block**

**Byte15-2: Reserved**

In each case, 0 is read.

Writing of data is not valid.

**Byte1-0: Card Key Version (CKV)**

Version of Card Key is stored here.

When reading data from this Block, only Service Attribute of specified Service Code is checked. An arbitrary value can be set in Service Number.

## 3.1.11CK

### Card Key Block (Block Number: 87h)

This Block stores Card Key. When the IC is shipped from the factory, a predetermined fixed value other than ALL_00h is stored in all chips. Rewrite of arbitrary data is possible. Allocation of data is as shown in Figure 3-13.

Data-write after setting System Block to the rewrite prevention state can be set to write protection (read-only) or to accept Write With MAC.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| CK1 | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| CK2 | | | | | | | |

**Figure 3-13: Card Key Block**

**Byte15-8: Card Key 2 (CK2)**

Card Key CK2 is stored here. Regardless of the value written in this way, in each case, 0 is read.

**Byte7-0: Card Key 1 (CK1)**

Card Key CK1 is stored here. Regardless of the value written in this way, in each case, 0 is read.

CK2 and CK1 are used as the encryption key to generate the session key. In each case, the value read from this Block is ALL_00h. Therefore, it is impossible to read and check the value written to this Block. The value stored in this Block is used to generate the MAC, however, so you can check and verify the result of data written to this Block by comparing the result of the newly-generated MAC with the value you expected. For details, see section 5.1“MAC generation procedure of MAC Block”.

## 3.1.12 MC

### Memory Configuration Block (Block Number: 88h)

This Block stores information such as “access permission for Block”, “settings to store NDEF data”, and “RF parameter”. When the IC is shipped from the factory, FFFFFF00FF0000…00h is stored. Rewriting of the data value is possible only for the first 13 Bytes. The setting based on the value written to this Block becomes valid after the next power on. Allocation of data is as shown in Figure 3-14.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| MC_SP_REG_ALL_RW | | MC_ALL | SYS_OP | RF_PRM | *1 | *2 | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| *3 | | *4 | | *5 | — | | |

*1 MC_CKCKV_W_MAC_A
*2 MC_SP_REG_R_RESTR
*3 MC_SP_REG_W_RESTR
*4 MC_SP_REG_W_MAC_A
*5 MC_STATE_W_MAC_A

**Figure 3-14: Memory Configuration Block**

**Byte15-13: Reserved**

In each case, 0 is read.

Writing of data is not valid.

**Byte12: Memory Config (MC_STATE_W_MAC_A)**

Set STATE Block to the Write With MAC attribute. Set MC_STATE_W_MAC_A = 01h to perform Read After Authentication or Write After Authentication. After setting each bit of this Byte to 1b, it cannot be returned to 0b.

01h: MAC is required for data-write.

00h: MAC is not required for data-write.

02h-FFh: Reserved

NOTE Set 1b to bits which correspond to Read After Authentication and Write After Authentication in MC_SP_REG_W_RESTR and MC_SP_REG_R_RESTR.

**Byte11-10: Memory Config (MC_SP_REG_W_MAC_A)**

Set Write With MAC for S_PAD0-S_PAD13 Block and REG Block. After setting each bit of these Bytes to 1b, it cannot be returned to 0b.

1b: MAC is required for data-write.

0b: MAC is not required for data-write.

**Byte9-8: Memory Config (MC_SP_REG_W_RESTR)**

Set Write After Authentication for S_PAD0-S_PAD13 Block and REG Block. When authentication is required, set bits which correspond to Blocks to be used in MC_SP_REG_W_REST and MC_STATE_W_MAC_A to 1b's. After setting each bit of these Bytes to 1b, it cannot be returned to 0b.

1b: Authentication is required for data-write.

0b: Authentication is not required for data-write.

NOTE Change b0 to 1b of MC_STATE_W_MAC_A if Write After Authentication needs to be used

**Byte7-6: Memory Config (MC_SP_REG_R_RESTR)**

Set Read After Authentication for S_PAD0-S_PAD13 Block and REG Block. When authentication is required, set bits which correspond to Blocks to be used in MC_SP_REG_W_REST and MC_STATE_W_MAC_A to 1b's. After setting each bit of these Bytes to 1b, it cannot be returned to 0b.

1b: Authentication is required for data-read.

0b: Authentication is not required for data-read.

NOTE Change b0 to 1b in MC_STATE_W_MAC_A if Read After Authentication needs to be used.

**Byte5: Memory Config (MC_CKCKV_W_MAC_A)**

Set Write With MAC for CK Block and CKV Block. To rewrite CK Block and CKV Block with MAC after the 1$^{st}$ issuance, set MC_CKCKV_W_MAC_A = 01h.

01h: MAC is required for data-write.

00h: MAC is not required for data-write.

02h-FFh: Reserved

**Byte4: RF Parameter (RF_PRM)**

The data in the lower 3 bits of this Byte is set as the RF parameter. When writing data to this Block, make sure you write only the value (i.e., 07h) specified in the procedure of issuance. Do not write any value other than this.

**Byte3: System Option (SYS_OP)**

Perform the setting to store NDEF data. When the value of this Byte is 01h, FeliCa Lite-S is compatible with NDEF, and returns a response to the Polling command in which System Code of 88B4h or 12FCh is specified. If compatibility with NDEF is unnecessary, set this Byte to 00h. With this setting, FeliCa Lite-S returns a response only to the Polling command in which System Code of 88B4h is specified.

01h: Compatible with NDEF

00h: Incompatible with NDEF

02h-FFh: Reserved

**Byte2: Memory Config (MC_ALL)**

Set System Block (Block Number: 82h, 83h, 84h, 86h, 87h) and the access permission for MC [2] - [5], and reset WCNT. When the value of this Byte is FFh, the access permission to System Block and to MC [2] - [5] becomes RW. When the value of this Byte is other than FFh, the access permission becomes RO. When writing data to this Byte, write FFh or 00h. When the electrical power is shut off after 00h was written, any change to this value becomes impossible.

FFh: RW

00h: RO

01h - FFh: Reserved

**Byte1-0: Memory Config (MC_SP_REG_ALL_RW)**

Set the access permission for scratch pad, subtraction register, MC [0] - [1], and MC [6] - [12]. When the electrical power is shut off after 0b was written to each of bits, any change to this value becomes impossible.

1b: RW

0b: RO

You can set-up the access permission in only two steps, as follows:

1) Set the access permission for System Block using MC_ALL.
2) Set the access permission for User Block using MC_SP_REG_ALL_RW, MC_SP_REG_R_RESTR, MC_SP_REG_W_RESTR, MC_SP_REG_W_MAC_A, and MC_STATE_W_MAC_A.

When the IC is shipped from the factory, RW permission is set for each Block except MAC Block, SYS_C Block, WCNT Block, and CRC_CHECK Block.

Table 3-2 shows the bit assignment of MC Block.

### Table 3-2: Bit assignment of MC Block

| Byte | Bit | Attribute and security setting | Polarity |
|---|---|---|---|
| MC[0] | bit0 | Setting of read and write for S_PAD0 | 1b:RW<br>0b:RO |
| | bit1 | S_PAD1 | |
| | bit2 | S_PAD2 | |
| | bit3 | S_PAD3 | |
| | bit4 | S_PAD4 | |
| | bit5 | S_PAD5 | |
| | bit6 | S_PAD6 | |
| | bit7 | S_PAD7 | |
| MC[1] | bit0 | S_PAD8 | |
| | bit1 | S_PAD9 | |
| | bit2 | S_PAD10 | |
| | bit3 | S_PAD11 | |
| | bit4 | S_PAD12 | |
| | bit5 | S_PAD13 | |
| | bit6 | REG | |
| | bit7 | MC[0]-[1], MC[6]-[12] | |
| MC[2] | bit0 | Setting of read and write for System Block and MC[2]-[5] | FFh: RW<br>Other than FFh: RO |
| | bit1 | | |
| | bit2 | | |
| | bit3 | | |
| | bit4 | | |
| | bit5 | | |
| | bit6 | | |
| | bit7 | | |
| MC[3] | bit0 | NDEF setting | 01h: Compatible with NDEF<br><br>00h: Incompatible with NDEF<br><br>02h – FFh: Prohibited |
| | bit1 | | |
| | bit2 | | |
| | bit3 | | |
| | bit4 | | |
| | bit5 | | |
| | bit6 | | |
| | bit7 | | |

| Byte | Bit | Attribute and security setting | Polarity |
|---|---|---|---|
| MC[6] | bit0 | Setting of Read After Authentication for S_PAD0 | Data-read requires:<br>1b: Authentication<br>0b: No authentication |
| | bit1 | S_PAD1 | |
| | bit2 | S_PAD2 | |
| | bit3 | S_PAD3 | |
| | bit4 | S_PAD4 | |
| | bit5 | S_PAD5 | |
| | bit6 | S_PAD6 | |
| | bit7 | S_PAD7 | |
| MC[7] | bit0 | S_PAD8 | |
| | bit1 | S_PAD9 | |
| | bit2 | S_PAD10 | |
| | bit3 | S_PAD11 | |
| | bit4 | S_PAD12 | |
| | bit5 | S_PAD13 | |
| | bit6 | REG | |
| | bit7 | - | - |
| MC[8] | bit0 | Setting of Write After Authentication for S_PAD0 | Data-write requires:<br>1b: Authentication<br>0b: No authentication |
| | bit1 | S_PAD1 | |
| | bit2 | S_PAD2 | |
| | bit3 | S_PAD3 | |
| | bit4 | S_PAD4 | |
| | bit5 | S_PAD5 | |
| | bit6 | S_PAD6 | |
| | bit7 | S_PAD7 | |
| MC[9] | bit0 | S_PAD8 | |
| | bit1 | S_PAD9 | |
| | bit2 | S_PAD10 | |
| | bit3 | S_PAD11 | |
| | bit4 | S_PAD12 | |
| | bit5 | S_PAD13 | |
| | bit6 | REG | |
| | bit7 | - | - |

| Byte | Bit | Attribute and security setting | Polarity |
|---|---|---|---|
| MC[4] | bit0 | RF parameter | Set 07h |
| | bit1 | | |
| | bit2 | | |
| | bit3 | - | - |
| | bit4 | - | - |
| | bit5 | - | - |
| | bit6 | - | - |
| | bit7 | - | - |
| MC[5] | bit0 | Setting of Write With MAC for CK and CKV | *1 |
| | bit1 | - | - |
| | bit2 | - | - |
| | bit3 | - | - |
| | bit4 | - | - |
| | bit5 | - | - |
| | bit6 | - | - |
| | bit7 | - | - |

*1 After setting System Block to the rewrite prevention state

1b: Rewrite is possible using Write With MAC

0b: Rewrite is impossible

*2 Data-write requires

1b: MAC

0b: no MAC

| Byte | Bit | Attribute and security setting | Polarity |
|---|---|---|---|
| MC[10] | bit0 | Setting of Write With MAC for S_PAD0 | Data-write requires:<br>1b: MAC<br>0b: No MAC |
| | bit1 | S_PAD1 | |
| | bit2 | S_PAD2 | |
| | bit3 | S_PAD3 | |
| | bit4 | S_PAD4 | |
| | bit5 | S_PAD5 | |
| | bit6 | S_PAD6 | |
| | bit7 | S_PAD7 | |
| MC[11] | bit0 | S_PAD8 | |
| | bit1 | S_PAD9 | |
| | bit2 | S_PAD10 | |
| | bit3 | S_PAD11 | |
| | bit4 | S_PAD12 | |
| | bit5 | S_PAD13 | |
| | bit6 | REG | |
| | bit7 | - | - |
| MC[12] | bit0 | Setting of Write With MAC for STATE | *2 |
| | bit1 | - | - |
| | bit2 | - | - |
| | bit3 | - | - |
| | bit4 | - | - |
| | bit5 | - | - |
| | bit6 | - | - |
| | bit7 | - | - |

## 3.1.13 WCNT

### WCNT Block (Block Number: 90h)

This Block stores the value of the write counter. When the IC is shipped from the factory, 00h FEh FFh is stored in WCNT [0] - [2]. This Block is read-only. Allocation of data is as shown in Figure 3-15.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| WCNT | | | – | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| – | | | | | | | |

**Figure 3-15: Write Counter Block**

**Byte15-3: Reserved**

In each case, 0 is read.

**Byte2-0: WCNT**

The value of the write counter is stored.

When the IC is shipped from the factory, WCNT [0] - [2] is set to 00h FEh FFh. Every time data-write occurs to S_PAD0-S_PAD13 Block, REG Block, ID Block, SER_C Block, CKV Block, CK Block, MC Block and STATE Block (when MC_STATE_W_MAC_A = 01h), the value is incremented by 1. After the value reaches FFFFFFh, it is not incremented even if data-write occurs.

To set System Block to the rewrite prevention state in the 1st issuance, set MC [2] to 00h. At this time, WCNT is reset to 000000h.

After that, every time data-write occurs to S_PAD0-S_PAD13 Block, REG Block, CKV Block, CK Block, MC Block and STATE Block (when STATE_W_MAC_A = 01h), the value is incremented by 1. After the value reaches FFFE00h, Write With MAC becomes impossible. Simple Write, however, is possible. The WCNT is not incremented even if data-write occurs.

Table 3-3 provides a summary of WCNT action. “After 1st issuance task is completed” represents time after setting MC [2] to 00h and then restarting the card. “2nd issuance is working” represents time when the setting of MC [0], [1], [6], [7], [8], [9], [10], [11], [12] is being performed. “being used by end users” represents time after the 2nd issuance is completed. In this table, the Write Counter value is notated in Little Endian format.

**Table 3-3: Action of WCNT**

| Life cycle | Write Counter value (Little Endian) | WCNT action during Simple Write and Write With MAC | Simple Write (SF1 SF2) | Write With MAC (SF1 SF2) | Read (SF1 SF2) |
|---|---|---|---|---|---|
| 0th (factory default) | 00h FEh FFh | - | Normal (00h 00h) | Normal (00h 00h) | Normal (00h 00h) |
| 1st issuance is working | 00h FEh FFh - FEh FFh FFh | WCNT is incremented by 1 | Normal (00h 00h) | Normal (00h 00h) | Normal (00h 00h) |
| | FFh FFh FFh | Clip (WCNT=FFFFFFh) | Normal (00h 00h) | Error (02h B2h) | Normal (00h 00h) |
| After 1st issuance task is completed | 00h 00h 00h | - | Normal (00h 00h) | Normal (00h 00h) | Normal (00h 00h) |
| 2nd issuance is working or being used by end users | 00h 00h 00h - 10h 27h 00h (002710h=10000d) | WCNT is incremented by 1 | Normal (00h 00h) | Normal (00h 00h) | Normal (00h 00h) |
| | 11h 27h 00h - FFh FDh FFh | | Warning (FFh 71h) | Warning (FFh 71h) | Normal (00h 00h) |
| | 00h FEh FFh | Clip (WCNT=FFFE00h) | Warning (FFh 71h) | Error (02h B2h) | Normal (00h 00h) |

## 3.1.14 MAC_A

### MAC_A Block (Block Number: 91h)

This Block is used for reading or writing the value of MAC. Allocation of data is as shown in Figure 3-16 and Figure 3-17.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| MAC_A | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| – | | | | | | | |

**Figure 3-16: MAC_A Block (data-read)**

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| MAC_A | | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| WCNT (Little Endian) | | | – | | | | |

**Figure 3-17: MAC_A Block (data-write)**

**Byte15-11: Reserved**

In each case, 0 is read. These Bytes cannot be rewritten.

**Byte10-8: WCNT**

In each case, 0 is read. Write the value of WCNT.

**Byte7-0: MAC_A**

Read or write the value of the result of the MAC calculation.

## 3.1.15 STATE

### STATE Block (Block Number: 92h)

This Block is used for External Authentication and the Polling Disable function. Data written to this Block is lost when the electric power is shut off. When the IC is booted up, ALL_00h is read in this Block.

Allocation of data is as shown in Figure 3-18.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| EXT AUTH | – | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| POLL _DIS | – | | | | | | |

**Figure 3-18: STATE Block**

**Byte15-9: Reserved**

In each case, 0 is read.

Writing of data is not valid.

**Byte8: POLL_DIS**

This Byte indicates whether to respond to the Polling command.

01h: Does not respond

00h: Respond

02h-FFh: Reserved

**Byte7-1: Reserved**

In each case, 0 is read.

Writing of data is not valid.

**Byte0: EXT_AUTH**

This Byte indicates the state before and after authentication.

If this Byte indicates the state after authentication, each Block that is set to the Read After Authentication attribute and/or the Write After Authentication attribute accepts the Read Without Encryption command and/or the Write Without Encryption command.

01h: After authentication

00h: Before authentication

02h-FFh: Reserved

## 3.1.16 CRC_CHECK

### CRC_CHECK Block (Block Number: A0h)

This Block is used for reading the result of verification of the CRC of each Block. The CRC is calculated from the data of each Block saved in non-volatile memory. The verification of the CRC is performed only once during restart. This Block is read-only. Allocation of data is as shown in Figure 3-19.

| [0] | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|
| CRC_CHECK | – | | | | | | |
| [8] | [9] | [10] | [11] | [12] | [13] | [14] | [15] |
| – | | | | | | | |

**Figure 3-19: CRC_CHECK Block**

**Byte15-1: Reserved**

In each case, 0 is read.

**Byte0: CRC_CHECK**

Using this Byte, you can read the result of verification of the CRC.

00h: The CRC calculated from data of all Blocks matches the CRC stored.
FFh: The CRC calculated from data of all Blocks does not match the CRC tored.
01h – FEh: Reserved

# 3.2 Service

"Service" is a group of Blocks located on the file system. Service provides access control to the Blocks so grouped.

All access to each Block is performed by using "Service". In FeliCa Lite-S products, one Service is registered for each for "Read/Write Access" and "Read Only Access" at the time of delivery. By using these two Services, access to any Block can be achieved.

To access Block under management of Service, first identify Service with a code of 2 Bytes known as Service Code. Then, by using a 1-Byte number known as Block Number, specify Block located in the range under management of Service specified by Service Code. Block Number starts from zero (0) within Service.

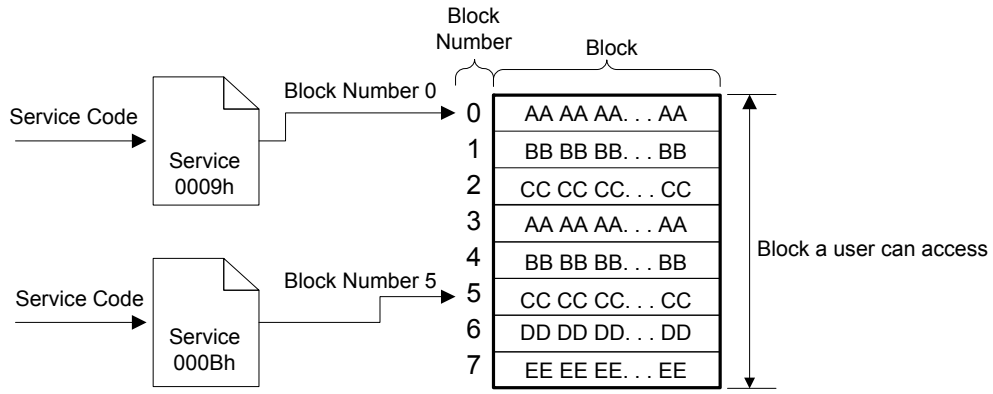



**Figure 3-20: Image of access to Block by Service**

There are three kinds of Service (Random, Cyclic, and Purse) in FeliCa Standard. Only Random Service is available for FeliCa Lite-S.

### <Service Code>

Service Code is determined by Service Number and Service Attribute. Configuration and format of Service Code is shown in Figure 3-21.

Example:

1) Read/Write Access: Service Code of Random Service
   (0000 0000 00**00 1001**) b = 0009h
2) Read Only Access: Service Code of Random Service
   (0000 0000 00**00 1011**) b = 000Bh

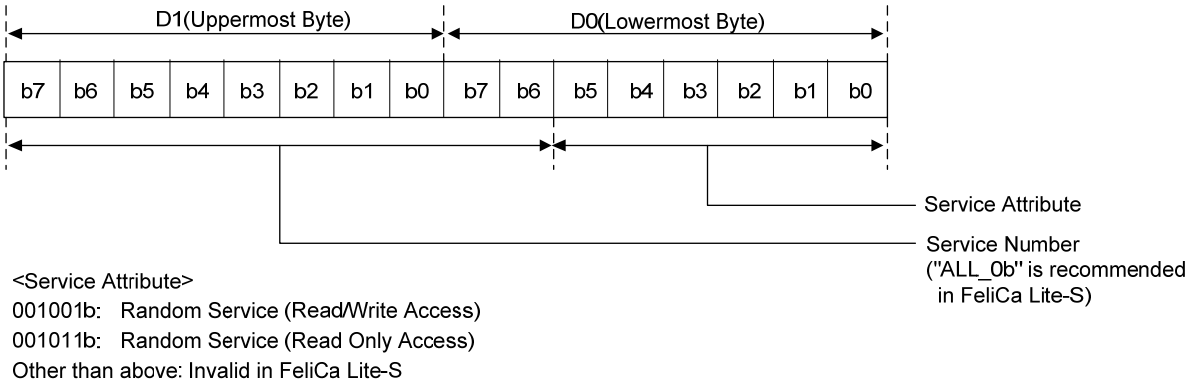



**Figure 3-21: Configuration of Service Code**

### <Service Attribute>

In Random Service, the following two variants of Service Attribute are provided:

| Service Attribute | Description |
|---|---|
| Read/Write Access | Read and write of data are possible from and to RW Permission Block. |
| Read Only Access | Read of data is possible from RO Permission Block and RW Permission Block. |

### <Configuration of Block>

Any data can be stored in Block.

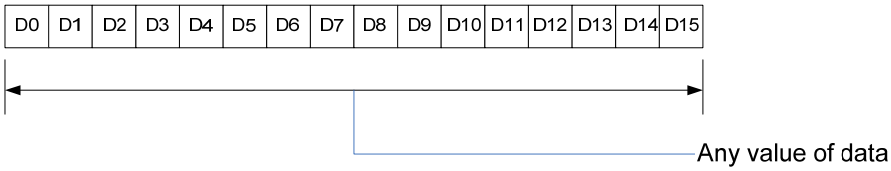



**Figure 3-22: Block Data of random access**

### <How to specify Block>

You can specify Block with arbitrary Block Number.

## 3.3 Area

There is no concept of Area in FeliCa Lite-S products.

# 3.4 System

System is the normative unit to be handled as a single logical card. FeliCa Lite-S has no System Separation function. Therefore, only one System exists on a single card.

### <System Code>

System is provided with a code of 2 Bytes known as System Code. The Reader/Writer uses System Code to identify the card (i.e., System). System Code of FeliCa Lite-S is 88B4h, which can be used to identify FeliCa Lite-S.

When identifying a card, the Reader/Writer is required to call (i.e., poll) a large indefinite number of cards with the Polling command. In this procedure, System Code is specified as a parameter of the Polling command. System returns a response only when each instance of System Code matches each other, as a preliminary step in the anti-collision process.

In FeliCa Lite-S, System Code cannot be changed, but System can be set to respond to the Polling command in which the value (i.e., 12FCh) is specified as the parameter. For details, see section 3.1.12“MC”.

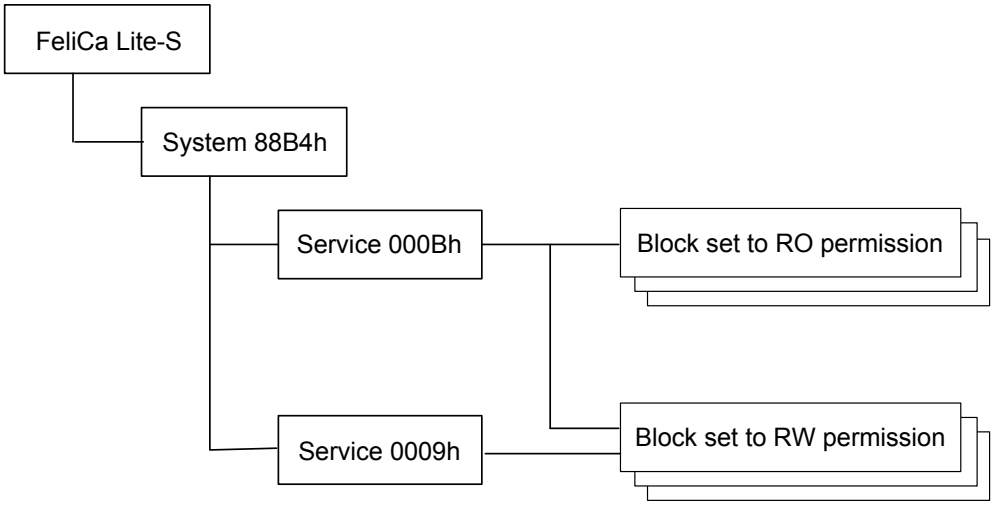



**Figure 3-23: Concept diagram of file system containing System**

# 3.5 Logical hierarchical structure

In FeliCa Lite-S products, there is no concept of the hierarchical structure of Area and Service in FeliCa Standard products.

# 3.6 Protection of data

## 3.6.1 Data protection function against power interruption

It is guaranteed that the update of data located in non-volatile memory with a Write Without Encryption command certainly results in either "totally updated" or "nothing updated". This is the function to maintain integrity of the data in non-volatile memory even if the update process was interrupted by shutting off the electrical power to the IC. Data writing to Blocks mentioned below is handled as valid only when all the data is written successfully. If data writing was interrupted by shutting-off of the electrical power to IC, the data-write operation is terminated and the data stored before such data writing is maintained.

This data protection function applies to the S_PAD0-13 Block, REG Block, ID Block, SER_C Block, CKV Block, CK Block, and MC Block.

In FeliCa Lite-S, data can be written to each Block by using only a single Write Without Encryption command.

# 4 Commands

This chapter describes the specifications of FeliCa Lite-S commands.

## 4.1 Acquisition and identification of card

This section describes how to acquire and identify a card (i.e., System) from the Reader/Writer.

To acquire a card from the Reader/Writer, the Reader/Writer calls (i.e., polls) an indefinite number of cards using the Polling command. To specify a required card (i.e., System), the Reader/Writer uses System Code, as described in section 3.4“System” of this document.

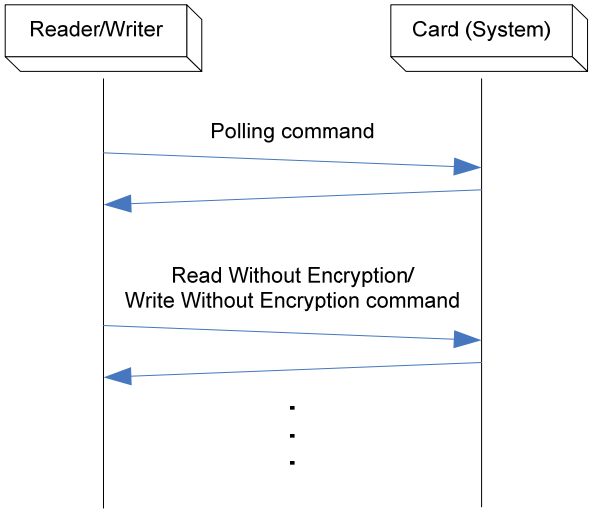
When polling is performed with the Polling command, cards return IDm and PMm as the response to the command. After this, communication with only a specific card (i.e., System) becomes possible using the acquired IDm.

To identify the destination card of communication using IDm, see section 2.3.3“Anti-collision process”. For details of the Polling command, see section 4.4 “Command specifications”.

# 4.2 Access to Block

This section describes how to read Block from and write Block to the FeliCa Lite-S file system.

To access Block, the Read Without Encryption command and the Write Without Encryption command are used.




1. Acquisition of a card (System)
   Transmit the Polling command to acquire IDm as card identification information.

2. Read and write of Block Data
   Transmit the Read Without Encryption command or the Write Without Encryption command, specifying Service Code List and Block List.
   For the Write Without Encryption command, also specify Block Data to write.

**Figure 4-1: Example of command sequence**

To access Block it is necessary to specify Service by using Service Code, and then, specify Block by using Block Number. To specify them using commands, use data structures named Service Code List and Block List.

## 4.2.1 Block List and Block List Element

Block List is used to identify the value of Service to be the target of access. In Block List, elements of data, each known as Block List Element, are enumerated. Figure 4-2 to Figure 4-4 show the configurations of Block List and Block List Element.

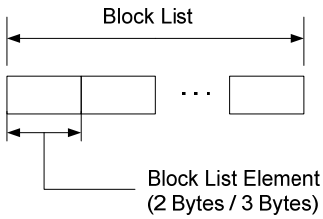



**Figure 4-2: Block List**

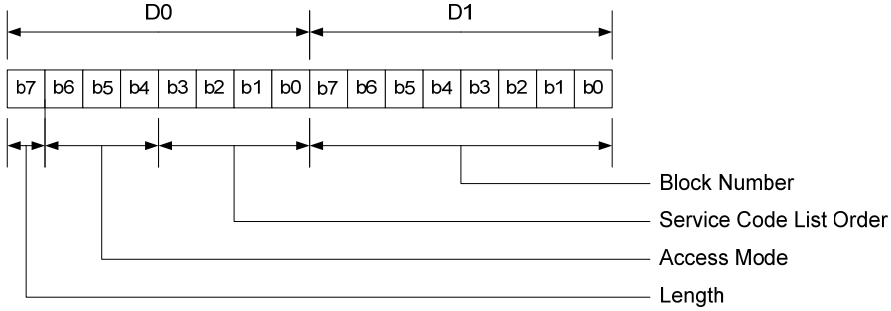



**Figure 4-3: 2-Byte Block List Element**

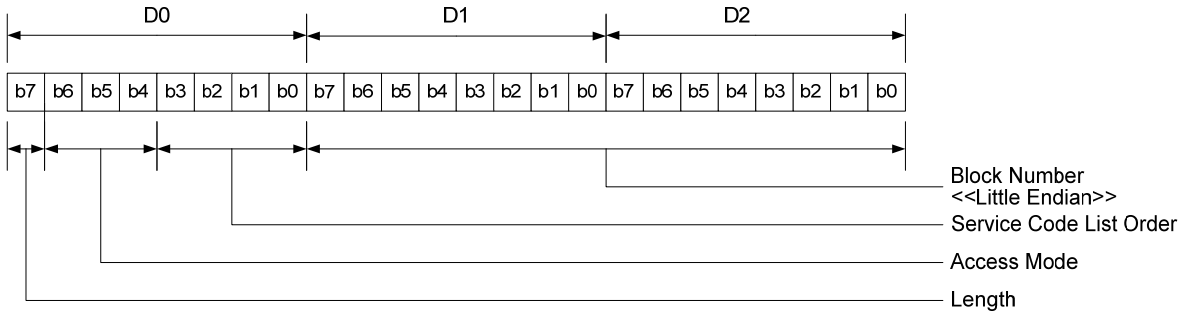



**Figure 4-4: 3-Byte Block List Element**

In FeliCa Lite-S, Block List Element is used only to specify Block Number of the target destination. Therefore, fixed values are set to Access Mode and Service Code List Order.

Block Number can be specified with data of 1 Byte, so each Block is accessible with 2-Byte Block List Element. Also possible, however, is to use 3-Byte Block List Element to access Block.

Additionally, Block Data to be written to Service are enumerated in parameters of the Write Without Encryption command, independently of Block List. For the procedure to store Block List and Block Data in the command packet, see section 4.4 “Command specifications”.

The following contents shall be specified to Block List Element with the format as shown in Figure 4-3 and Figure 4-4:

- Length
  Specify whether Block List Element is 2 Bytes or 3 Bytes long.
    1b: Block List Element consists of 2 Bytes.
    0b: Block List Element consists of 3 Bytes.
- Access Mode
  In FeliCa Lite-S, in each case, 000b is specified.
- Service Code List Order
  In FeliCa Lite-S, in each case, 0000b is specified.
- Block Number
  Specify Block Number you want to access.

For settings of Block List and Block List Element, see also section 4.2.2 “Example of Block List settings”.

## 4.2.2 Example of Block List settings

This section shows some of examples of command packet used when accessing Block.

**Example 1:**

Content of access
  Read “scratch pad 5” and “MAC Block” simultaneously.
Information for creation of the command packet
  Command Code: 06h (Read Without Encryption)
  IDm: 8-Byte value acquired by the Polling command
  Number of Service: 1
  Service Code: 000Bh
    Service Attribute: 001011b (RO Service)
  Number of Block: 2
  Block List: 80h 05h 80h 81h
    First Block List Element: (80h 05h)
      Length: 2 Bytes
      Block Number: 05h (Block Number of scratch pad 5)
    Second Block List Element: (80h 81h)
      Length: 2 Bytes
      Block Number: 81h (Block Number of MAC Block)

Figure 4-5 shows the content of the new command packet, based on the information in this example. Be aware that the entries in Service Code List are sorted in Little Endian order.

| Command Code | IDm | | | | | | | | Number of Service | Service Code List | | Number of Block | Block List | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 06h | IDm[0] | IDm[1] | IDm[2] | IDm[3] | IDm[4] | IDm[5] | IDm[6] | IDm[7] | 01h | 0Bh | 00h | 02h | 80h | 05h | 80h | 81h |

**Figure 4-5: Example of command packet for Block read**

### Example 2:

Content of access

Write data to scratch pad 13.

Information for creation of the command packet

Command Code: 08h (Write Without Encryption)

IDm: 8-Byte value acquired by the Polling command

Number of Service: 1

Service Code: 0009h

Service Attribute: 001001b (RW Service)

Number of Block: 1

Block List: 80h 0Dh

Length: 2 Bytes

Block Number: 0Dh (Block Number of scratch pad 13)

Figure 4-6: Example of command packet for Block write shows the content of the new command packet, based on the information in this example.

| Command Code | IDm | | | | | | | | Number of Service | Service Code List | | Number of Block | Block List | | Block Data |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 08h | IDm[0] | IDm[1] | IDm[2] | IDm[3] | IDm[4] | IDm[5] | IDm[6] | IDm[7] | 01h | 09h | 00h | 01h | 80h | 0Dh | Write data (16 Bytes) |

**Figure 4-6: Example of command packet for Block write**

# 4.3 Mode transition

In FeliCa Lite-S products, there is no concept of Mode transition, as defined in FeliCa Standard. Instead, FeliCa Lite-S has its own concept of Mode, as shown in Figure 4-7.

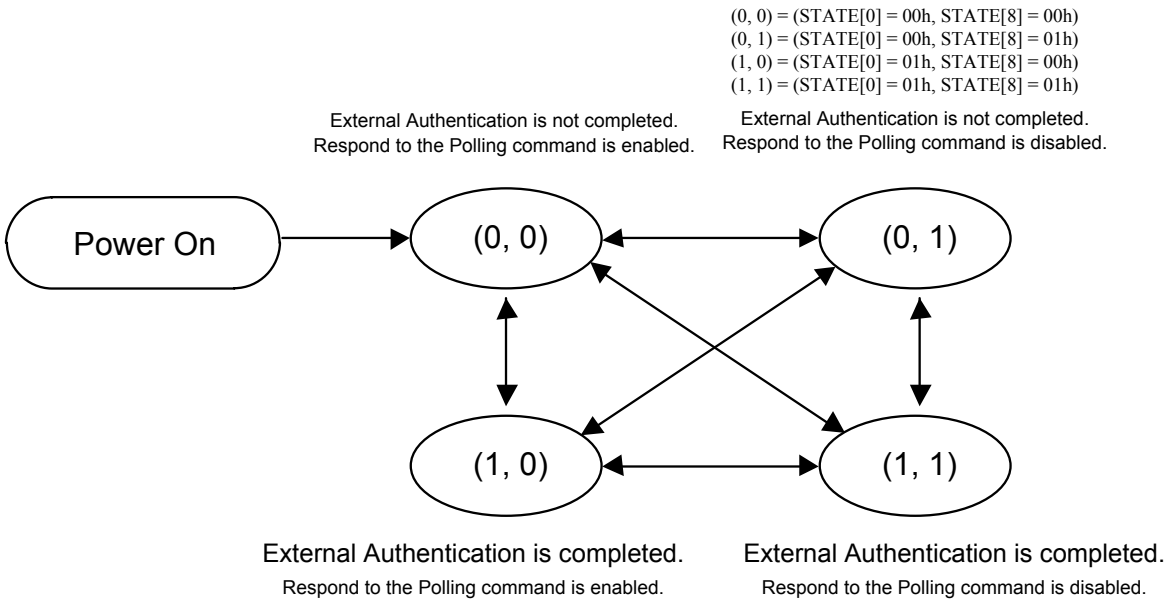



**Figure 4-7: Concept diagram of Mode in FeliCa Lite-S**

# 4.4 Command specifications

## 4.4.1 Structure of description

Each command interface is described in the following way:

**<Summary>**

Summarizes the functions of each command.

**<Requirement for returning a response>**

Describes the conditions under which a card (i.e., System) should return some type of response to a command transmitted from the Reader/Writer. If the conditions are not satisfied, the card will not respond.

**<Requirements for returning a response>**

Describes the conditions required for the successful completion of command execution. Only when all the requirements enumerated here are satisfied, does the command become successfully completed.

**<Packet structure>**

- **Command Packet Data**

  Describes the structure, parameter name, size, data, description, and other details (i.e., notes) of Command Packet Data at the time of command transmission (unit of size is represented in Bytes).

  Command Packet Data includes parameters (Service Code) for which Endian must be considered. For example, if notation ‹‹Little Endian›› exists in the “Note” column, the data must be set in Little Endian format.
- **Response Packet Data**

  Specifies the structure, parameter name, size, data, description, and other details (i.e., notes) of Packet Data at the time the response is returned (unit of size is represented in Bytes).

  Response Packet Data includes parameters (Service Code) for which Endian must be considered. For example, if notation ‹‹Little Endian›› exists in the “Note” column, the data must be set in Little Endian format.
- **Status Flag**

  Describes, for any command having Status Flag in its Response Packet Data, the value of the Status Flag and the nature of the error associated with that flag.

**<Special instruction>**

- Describes detailed information about the command, such as important notes to consider before using the command. Make sure that you check this description before using the command.

## 4.4.2 Polling

### <Summary>

- Use this command to acquire and identify a card.
- Acquisition of Manufacture ID (IDm) and Manufacture Parameter (PMm) is possible with this command.
- By specifying Request Code, you can acquire System Code or communication performance of System.
- By specifying Time Slot, you can designate the maximum number of Time Slots possible to return responses (see “<Special instruction>”).

### <Requirement in returning response>

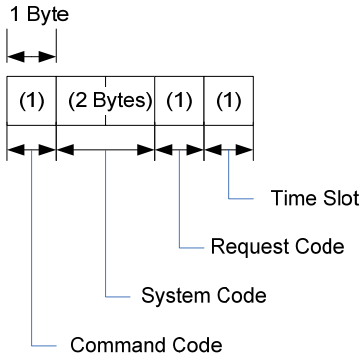
- The data length of the received packet shall be the correct data length for the Polling command.
- System specified by System Code shall exist in the card.
- Polling Disable function is set to respond to the Polling command.

### <Requirement for successful completion of command execution>

- All the requirements for returning a response shall be satisfied.

### <Packet structure>

### Command Packet Data

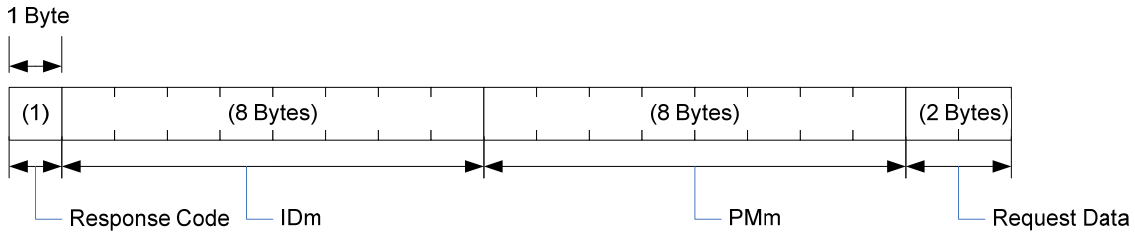



| Parameter name | Size | Data | Note |
|---|---|---|---|
| Command Code | 1 | 00h | |
| System Code | 2 | | You can specify a wildcard.<br>For details, see “<Special instruction>”. |
| Request Code | 1 | | Designation of Request Data.<br>00h: No request<br>01h: System Code request<br>02h: Communication performance request<br>other: RFU |

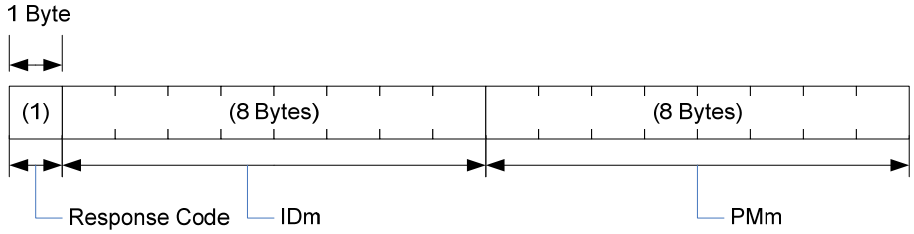
| Parameter name | Size | Data | Note |
|---|---|---|---|
| Time Slot | 1 | | Designation of maximum number of slots possible to respond.<br>(For details, see Table 4-1) |

### Response Packet Data

- The case when Request Data is returned:




- The case when Request Data is not returned:




| Parameter name | Size | Data | Note |
|---|---|---|---|
| Response Code | 1 | 01h | |
| IDm | 8 | | IDm of captured system |
| PMm | 8 | | |
| Request Data | 0 or 2 | | Data is returned only when Request Code of the command packet is not 00h, and corresponds to Request Data.<br>For details, see Figure 4-2, Figure 4-3, and “<Special instruction>”. |

**Table 4-1: Time Slot specifications**

| Time Slot | Maximum number of slots | Time Slot possible to respond |
|---|---|---|
| 00h | 1 | #0 |
| 01h | 2 | #0, #1 |
| 03h | 4 | #0, #1, #2, #3 |
| 07h | 8 | #0, #1, #2, #3, #4, #5, #6, #7 |
| 0Fh | 16 | #0, #1, #2, #3, #4, #5, #6, #7, #8, #9, #10, #11, #12, #13, #14, #15 |

**Table 4-2: Returned value to Request Data**

| Request Code | Request Data | Note |
|---|---|---|
| 00h: No request | None | Request Data is not returned. |
| 01h: System Code request | System Code | System Code of acquired System is returned. |
| 02h: Communication performance request | Communication performance | Communication performance (00h 83h) is returned. For the meaning of the value, see Table 4-3. |
| Other values: Reserved | None | Request Data is not returned |

**Table 4-3: Communication performance**

| D0 | D1 | | | | | | | | Description |
|---|---|---|---|---|---|---|---|---|---|
| | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 | |
| 00h<br>(other values are reserved) | - | - | - | - | - | - | - | x | 0b: 212kbps communication is impossible.<br>1b: 212kbps communication is possible. |
| | - | - | - | - | - | - | x | - | 0b: 424kbps communication is impossible.<br>1b: 424kbps communication is possible. |
| | - | - | - | - | - | 0 | - | - | 0b: 848kbps communication is impossible.<br>1b: 848kbps communication is possible (reserved) |
| | - | - | - | - | 0 | - | - | - | 0b: 1.6 Mbps communication is impossible.<br>1b: 1.6 Mbps communication is possible (reserved) |
| | - | 0 | 0 | 0 | - | - | - | - | Fixed value (other values are reserved) |
| | x | - | - | - | - | - | - | - | 0b: Communication rate automatic detection noncompliant.<br>1b: Communication rate automatic detection compliant. |

**<Special instruction>**

**Specifying a wildcard for System Code:**

- For System Code, you can specify a wildcard (FFh) for either the upper or lower 1 Byte, or for both the upper and lower Bytes. The Byte for which the wildcard is specified is regarded as an arbitrary value in the process of comparison with System Code of System existing in the card. If System Code is 88B4h, for example, the card returns a response when System Code of the Polling command is 88B4h (full matching), FFB4h (the upper 1 Byte is a wildcard), 88FFh (the lower 1 Byte is a wildcard), or FFFFh (both 2 Bytes are wildcards).
- By specifying a wildcard for both 2 Bytes (i.e., FFFFh), you can make a card return a response to the Polling command, regardless of its System Code.

### Specifying Time Slot

- Designation of Time Slot of the Polling command may be selected from (00h, 01h, 03h, 07h, or 0Fh). In this case, the number of Time Slots is (1, 2, 4, 8, or 16), respectively. If 00h is designated as Time Slot, for example, the timing at which the card returns a response is only once. Therefore, all the cards which can return a response start simultaneous returning of response. However, if 0Fh is designated, 16 timings are available for cards to return a response. It is expected that the probability of random timing selection increases and the probability of conflict decreases.

### On System Code returned in response to request for System Code

- When MC Block was set so that FeliCa Lite-S also returns a response to the Polling command that contains System Code of 12FCh, System Code of 88B4h is returned to the Polling command (with request for System Code) to which FFFFh is specified as System Code. System Code of 12FCh is returned as a response to the Polling commands that contain System Code of 12FCh, FFFCh, and 12FFh.

### 4.4.3 Read Without Encryption

**<Summary>**

- Use this command to read Block Data.

**<Requirement in returning response>**

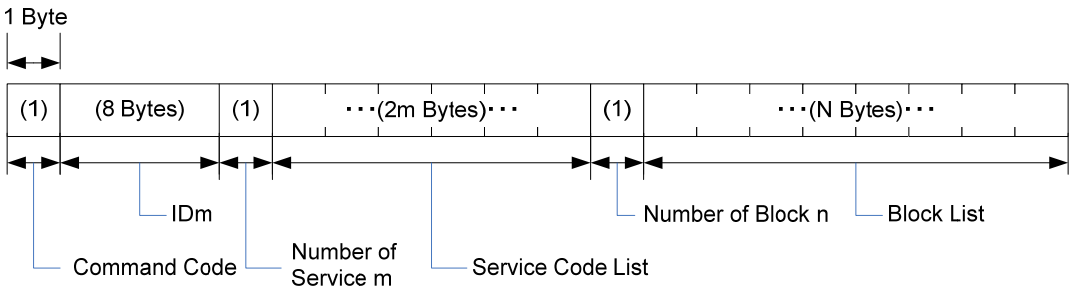
- The data length of the received packet shall be the correct data length of the Read Without Encryption command.

**<Requirement for successful completion of command execution>**

- Number of Service shall be 1.
- Number of Block shall be less than or equal to the maximum number of simultaneously-readable Blocks (4).
- Service specified by Service Code List shall exist in the card
- Each Block List Element shall satisfy the following conditions:
  - Value of Service Code List Order is 0000b.
  - Value of Access Mode is 000b.
  - Block Number is set to either 00h(S_PAD0)-0Eh(REG), 80h(RC)-88h(MC), 90h(WCNT)-92h(STATE), or A0h(CRC_CHECK).
  - When Service Attribute of Service Code is Read/Write Access, the access permission of the specified Block with Block Number is RW permission.
- MAC Block and MAC_A Block shall not be mixed.
- The security settings shall satisfy the following conditions:
  - When read Block is set to accept Read After Authentication, authentication is completed.
  - When MAC_A Block is specified, data-write to RC Block is completed.

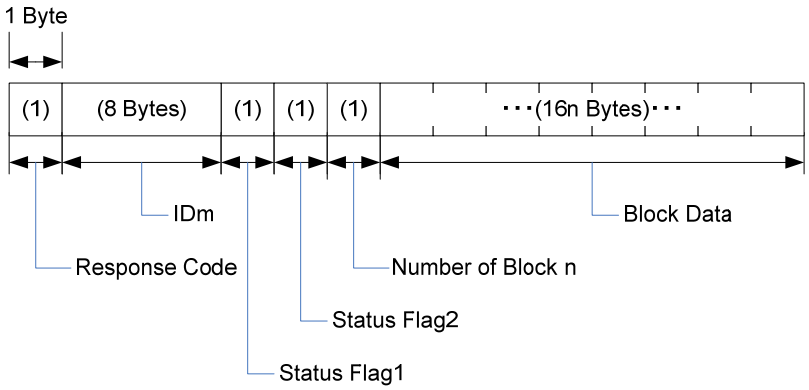
### <Packet structure>

### Command Packet Data




| Parameter name | Size | Data | Note |
|---|---|---|---|
| Command Code | 1 | 06h | |
| IDm | 8 | | |
| Number of Service | 1 | m | m=1 |
| Service Code List | 2m | | <<Little Endian>> |
| Number of Block | 1 | n | 1 ≤ n ≤ 4 |
| Block List | N | | See section 4.2.1“Block List and Block List Element” for Block List Element.<br>Possible to specify mixture of 2-Byte and 3-Byte Block List Elements.<br>2n ≤ N ≤ 3n |

## Response Packet Data




| Parameter name | Size | Data | Note |
|---|---|---|---|
| Response Code | 1 | 07h | |
| IDm | 8 | | |
| Status Flag1 | 1 | | See section 4.5“Status Flag” |
| Status Flag2 | 1 | | See section 4.5“Status Flag” |
| Number of Block | 1 | n | Provided only if Status Flag1 = 00h. |
| Block Data | 16n | | Provided only if Status Flag1 = 00h. |

### 4.4.4 Write Without Encryption

**<Summary>**

- Use this command to write Block Data.

**<Requirement in returning response>**

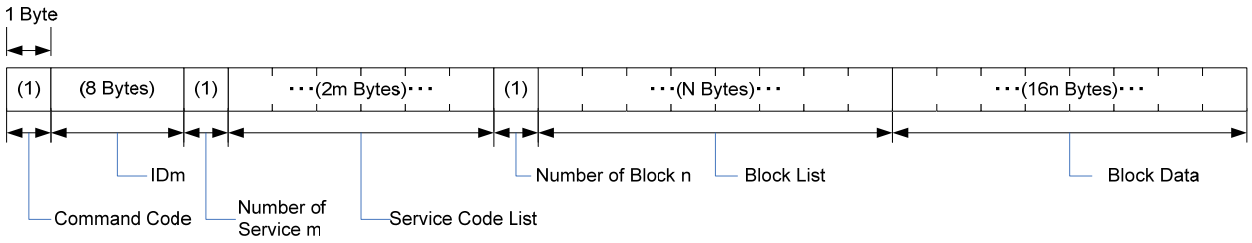
- The data length of the received packet shall be the correct data length of the Write Without Encryption command.

**<Requirement for successful completion of command execution>**

- Number of Service shall be 1.
- Number of Block shall be less than or equal to maximum number of simultaneously-writable Blocks (1 or 2).
- Service Code List specified by Service exist in the card.
- Each Block List Element shall satisfy the following conditions:
  - Value of Service Code List Order is 0000b.
  - Value of Access Mode is 000b.
  - If Number of Blocks is 1, Block Number is set to either 00h(S_PAD0)-0Eh(REG), 80h(RC), 82h(ID), 84(SER_C), 86h(CK), 87h(CKV), 88h(MC), or 92h(STATE), and access permission of the specified Block is RW permission.
  - If Number of Block is 2, the Block number of the first Block is set to either 00h(S_PAD0)-0Eh(REG), 86h(CK), 87h(CKV), or 92h(STATE), and access permission of the specified Block is RW permission.
  - If Number of Block is 2, the Block number of the second Block is set to 91h(STATE)
- The security settings shall satisfy the following conditions:
  - When MAC_A Block is specified, data-write to RC Block is completed.
  - When write Block is set to accept Write After Authentication, authentication is completed.
  - When write Block is set to accept Write With MAC, the MAC and WCNT values are correct.

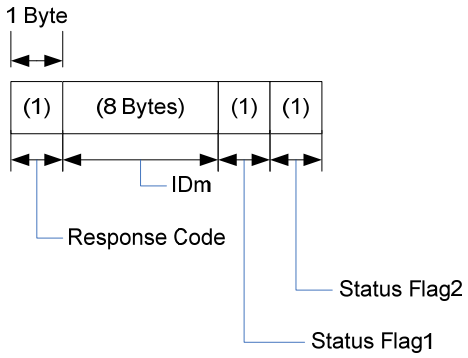
### <Packet structure>

### Command Packet Data




| Parameter name | Size | Data | Note |
|---|---|---|---|
| Command Code | 1 | 08h | |
| IDm | 8 | | |
| Number of Service | 1 | m | m=1 |
| Service Code List | 2m | | <<Little Endian>> |
| Number of Block | 1 | n | n = 1 or 2 |
| Block List | N | | See section 4.2.1"Block List and Block List Element" for Block List Element.<br><br>Possible to specify mixture of 2-Byte and 3-Byte Block List Elements.<br>$2n \leq N \leq 3n$ |
| Block Data | 16n | | |

## Response Packet Data




| Parameter name | Size | Data | Note |
|---|---|---|---|
| Response Code | 1 | 09h | |
| IDm | 8 | | |
| Status Flag1 | 1 | | See section 4.5“Status Flag”. |
| Status Flag2 | 1 | | See section 4.5“Status Flag”. |

# 4.5 Status Flag

Status Flag indicates the success or failure of the processing in a card and, if an error occurs during processing, provides details of the error.

Status Flag consists of Status Flag1 (1 Byte) and Status Flag2 (1 Byte), as follows:

## 4.5.1 Status Flag1

Status Flag1 indicates the success or failure of processing in the card, as well as the location of Block or of Service where an error occurred.

- 00h

  Indicates the successful completion of a command.
- FFh

  If the error is not related to Block List/Service Code List of the command packet, the card returns a response by setting Status Flag1 to FFh.
- XXh

  If an error occurs while processing the Read Without Encryption command or the Write Without Encryption command that includes Block List in the command packet, the card returns a response by setting a number in the list in Status Flag1, indicating the location of the error.

To indicate the location of the error occurrence with bit data, return Status Flag1 while setting the location in the following way:

Bit 0: First location of Block List

Bit 1: Second location of Block List

Bit 2: Third location of Block List

Bit 3: Fourth location of Block List

In this case, the value of each bit indicates the following:

0: No error

1: An error occurred

Data is checked from the lower location of the Block List.

The first Block that contains an error is set as the location of the error occurrence.

Example: When Status Flag 1 = 04h,

First location of Block List : no error

Second location of Block List : no error

Third location of Block List : error (Status Flag2 stores error information of this Block)

Fourth element of the Block List : not checked

## 4.5.2 Status Flag2

Status Flag2 indicates the detailed contents of an error. Status Flag2 is divided into two groups, i.e., the specifications common to FeliCa Standard card and the specifications unique to FeliCa Lite-S. System design engineers are requested to configure System using only the information common to FeliCa Standard card, and to avoid installation of information unique to FeliCa Lite-S. Use the specifications unique to FeliCa Lite-S card for debugging only.

### Specifications common to FeliCa Standard card (00h-7Fh)

**Table 4-4: Values and meanings of Status Flag2 (Specifications common to FeliCa Standard card)**

| Status Flag2 | Meaning |
|---|---|
| 00h | Indicates the successful completion of a command. |
| 70h | Memory error (fatal error). |
| 71h | The number of memory rewrites exceeds the upper limit (this is only a warning; data writing is performed as normal). |

### Specifications unique to FeliCa Lite-S (80h-FFh)

This Status Flag verifies the application.

Major error statuses are listed in Table 4-5: Major causes of error occurrence for the Read Without Encryption command and Table 4-6: Major causes of error occurrence for the Write Without Encryption command. In some error statuses, definition of the value and condition of error occurrence may differ from that of FeliCa Standard card. Therefore, it is not recommended to use these specifications to determine error occurrence during system operation.

Use these specifications for debugging purpose of application only.

### <Read Without Encryption command>

**Table 4-5: Major causes of error occurrence for the Read Without Encryption command**

| Cause of error occurrence | Response | SF1 | SF2 |
|---|---|---|---|
| LEN is not the predetermined value (i.e., does not satisfy the minimum command length). | No | – | – |
| IDm is inconsistent. | | | |
| Number of Service is not 01h. | Yes | FFh | A1h |
| Number of Block is not in the range of 01h to 04h. | | | A2h |
| LEN is not the predetermined value (i.e., the data length of Block List is not correct). | No | – | – |
| Access Mode of Block List is not 000b. | Yes | *1 | A7h |
| Service Code List Order of Block List is not 0000b. | | | A3h |
| Service Code of Service Code List is inconsistent. | | 01h | A6h |
| Service Attribute of Service Code of Service Code List is neither 001001b (RW) nor 001011b (RO). | | | |
| Block Number of Block List is not the predetermined value. | | *2 | A8h |
| Block identified with Block Number of Block List is RO permission area, and Service Attribute of Service Code is not 001011b (RO). | | | |
| Block Number (D2) is not 00h. | | | |
| MAC Block and MAC_A Block are mixed in Block List. | | | B0h |
| Block that requires authentication is read before authentication is completed. | | | B1h |
| Data-write to RC Block is not performed during data-read of MAC_A Block. | | | B2h |
| Data-read error. | | FFh | 70h |

*1, *2The value of the order in Block List at which an error was detected is set here.

### <Write Without Encryption command>

**Table 4-6: Major causes of error occurrence for the Write Without Encryption command**

| Cause of error occurrence | Response | SF1 | SF2 |
|---|---|---|---|
| LEN is not the predetermined value (i.e., does not satisfy the minimum command length). | No | – | – |
| IDm is inconsistent. | | | |
| Number of Service is not 01h. | Yes | FFh | A1h |
| Number of Block is neither 01h nor 02h. | | | A2h |
| LEN is not the predetermined value (i.e., the data length of Block List and Block Data is not correct). | No | – | – |
| Access Mode of Block List Element is not 000b. | Yes | *1 | A7h |
| Service Code List Order of Block List Element is not 0000b. | | | A3h |
| Service Code of Service Code List is inconsistent. | | 01h | A6h |
| Service Attribute of Service Code of Service Code List is not 001001b (RW). | | | |
| Block Number of Block List Element is not the predetermined value. | | *2 | A8h |
| An attempt was made to write to Block in the RO permission area, or to write without MAC to Block that requires MAC for data-write. | | 01h | A8h |
| Block Number (D2) is not 00h. | | *3 | A8h |
| The received REG[0]-[3] and REG[4]-[7] are larger than the written value. | | 01h | A9h |
| Block that requires authentication is written before authentication is completed. | | 01h | B1h |
| For Write With MAC, data-write to RC Block is not performed. | | 02h | B2h |
| For Write With MAC, the MAC value is inconsistent. | | | |
| For Write With MAC, WCNT is inconsistent. | | | |
| For retry of Write With MAC, the write data is inconsistent. | | | |
| Data-write count error for Write With MAC. | | | |
| Data-write error (i.e., an error occurred during CRC check after data-write.) | | FFh | 70h |
| The number of memory rewrites exceeds the predetermined value (this is only a warning; data writing is performed normally.) | | | 71h |

*1, *2, *3 The value of the order in Block List at which an error was detected is set here.

# 5 Security

This chapter describes (a) the MAC (Message Authentication Code) generating function supplied with FeliCa Lite-S, and (b) the security protocol that uses the MAC-generation function.

## 5.1 MAC generation procedure of MAC Block

### 5.1.1 MAC generation procedure for data-read

Features of the MAC-generation function supplied with FeliCa Lite-S are as follows:

- MAC generation uses the CBC mode of 2-key Triple DES.
- Generation of the key (i.e., the session key) for MAC generation is based on the key stored in IC and the random number specified by the Reader/Writer.
- MAC generation is possible for up to 3 Blocks.

MAC is generated by a 2-key Triple DES operation. The key (i.e., the session key) used for MAC generation is generated by a 2-key Triple DES operation based on a value stored in the card. Figure 5-1 shows the generation process of the session key.

The Byte order of data is reversed at the star symbol (☆).(CK[0]-[7]⇒CK[7]-[0])

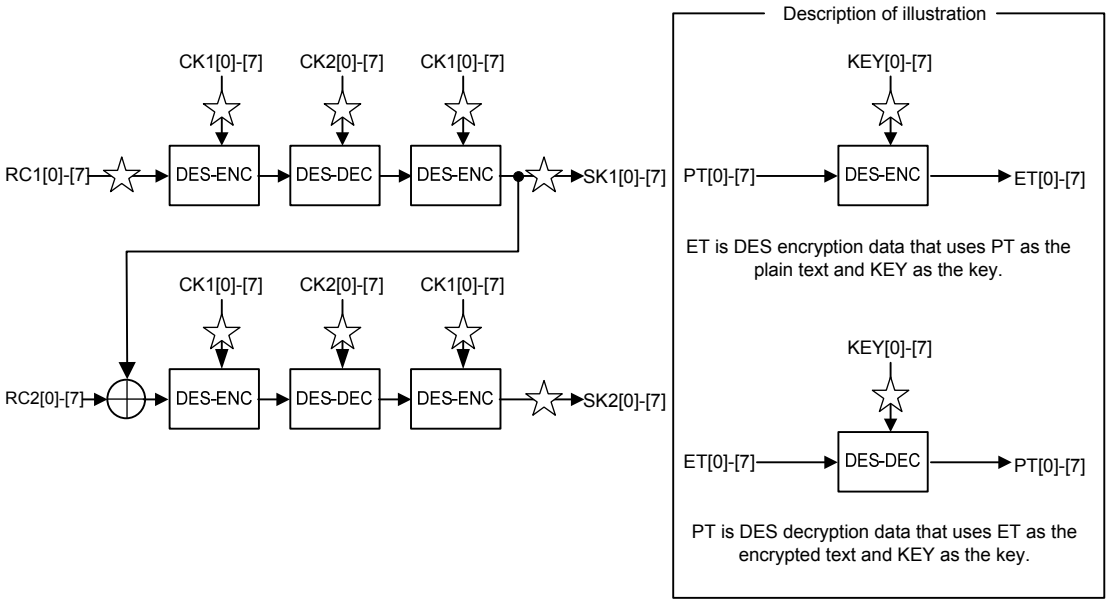



**Figure 5-1: Procedure to generate a session key**

The session keys (SK1, SK2) are generated by a 2-key Triple DES-CBC operation with IV (Initial Vector) 0 using Card Keys (CK1, CK2) as the key and Random Challenge (RC1, RC2) as the plain text. In FeliCa Lite-S, data of 8 Bytes each are processed so that the Byte with a suffix of a larger value is regarded as the upper Byte. The session key is used as the key for MAC generation.

Figure 5-2 shows the process of MAC generation for data-read.

MAC is generated by a 2-key Triple DES-CBC operation using session keys (SK1, SK2) as the key, data of read Block as the plain text, and Random Challenge (RC1) as the initial vector. MAC is generated for the data of Block that is read. (The data read from MAC Block, however, is not used in the MAC operation.) MAC generated for Block is used as the initial vector of MAC generation for the next Block. Figure 5-2 shows the process of MAC generation in the case where 2 Blocks are read.

The Byte order of data is reversed at the star symbol (☆). (CK[0]-[7]⇒CK[7]-[0])

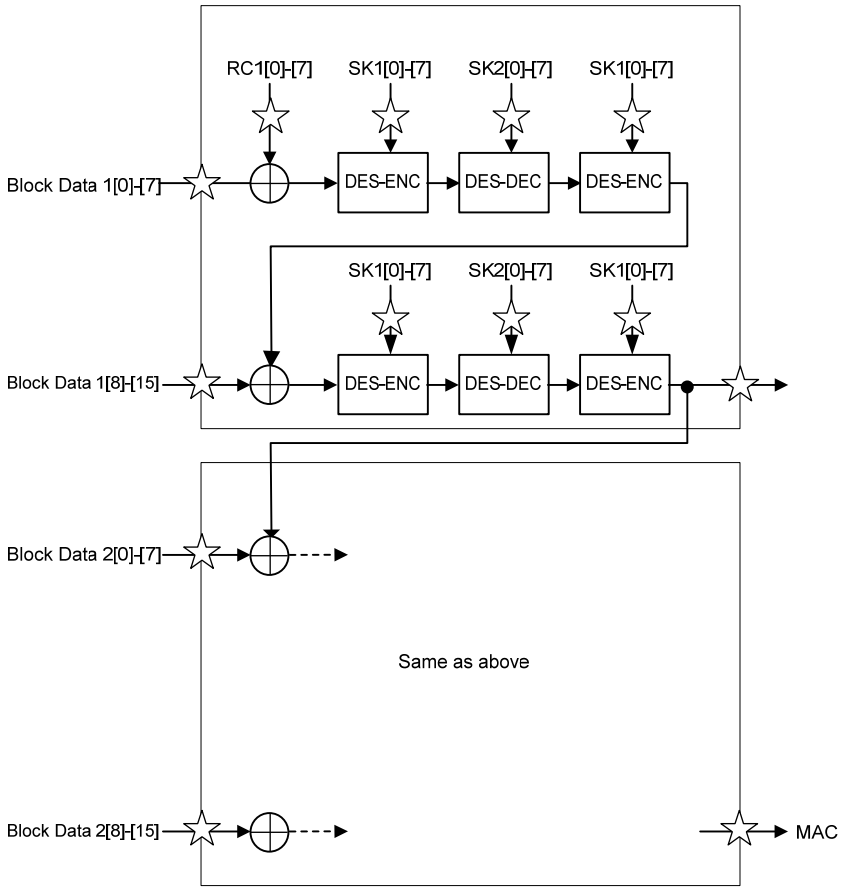



**Figure 5-2: MAC-generation procedure for data-read (MAC Block)**

## 5.1.2 MAC generation procedure for data-write

MAC Block does not support data-write.

# 5.2 MAC generation procedure of MAC_A Block

## 5.2.1 MAC generation procedure for data-read

Features of the MAC-generation function supplied with FeliCa Lite-S are as follows:

- MAC generation uses the CBC mode of 2-key Triple DES.
- Generation of the key (i.e., the session key) for MAC generation is based on the key stored in IC and an arbitrary value specified by the Reader/Writer.
- MAC generation is possible for up to 3 Blocks.
- Manipulation of data and Block Number of read blocks can be detected.

For MAC_A Block, the session key is generated in the same way as for MAC Block, as shown in Figure 5-1. The data read from MAC_A Block is not used in the MAC operation as it is with MAC Block. The procedure for calculating the MAC value, however, is different from that of MAC Block. For MAC_A Block, the MAC value that contains the data and Block Number of Block to be read is calculated.

Figure 5-3 shows the process of MAC generation for data-read.

The Byte order of data is reversed at the star symbol (☆). (CK[0]-[7]⇒CK[7]-[0])

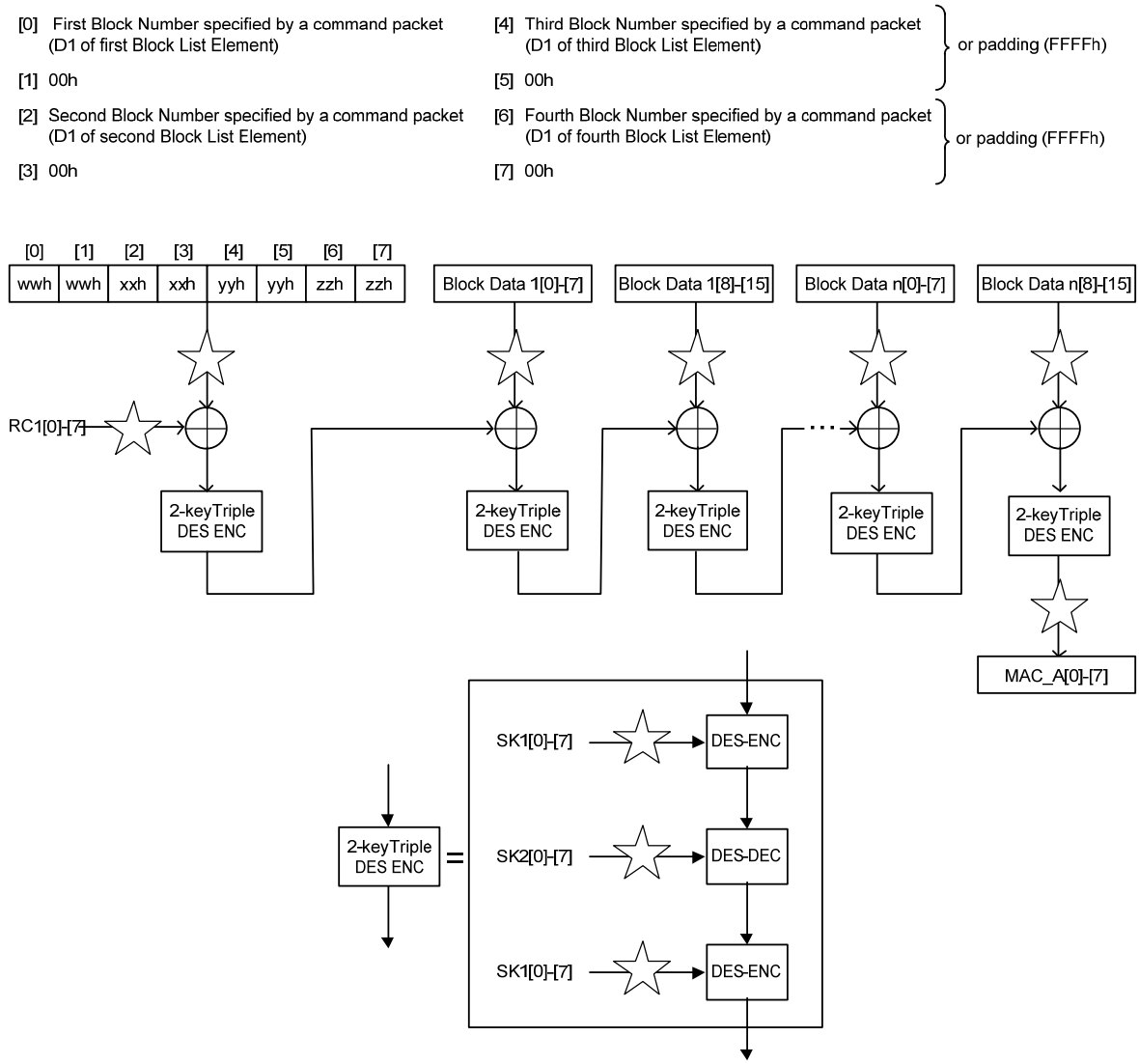



**Figure 5-3: MAC generation procedure for data-read (MAC_A Block)**

## 5.2.2 MAC generation procedure for data-write

FeliCa Lite-S is equipped with the MAC-generation function for data-write. Features are as follows:

- MAC generation uses the CBC mode of 2-key Triple DES.
- Generation of the key (i.e., the session key) for MAC generation is based on the key stored in IC and an arbitrary value specified by the Reader/Writer.
- MAC generation is possible for up to 1 Block.
- Manipulation of data and Block Number of write blocks can be detected.

The session key is generated in the way shown in Figure 5-1.

Figure 5-4 shows the MAC generation procedure for data-write. The value which is written to MAC_A Block by the Reader/Writer is compared with the generated MAC.

The Byte order of data is reversed at the star symbol (☆). (CK[0]-[7]⇒CK[7]-[0])

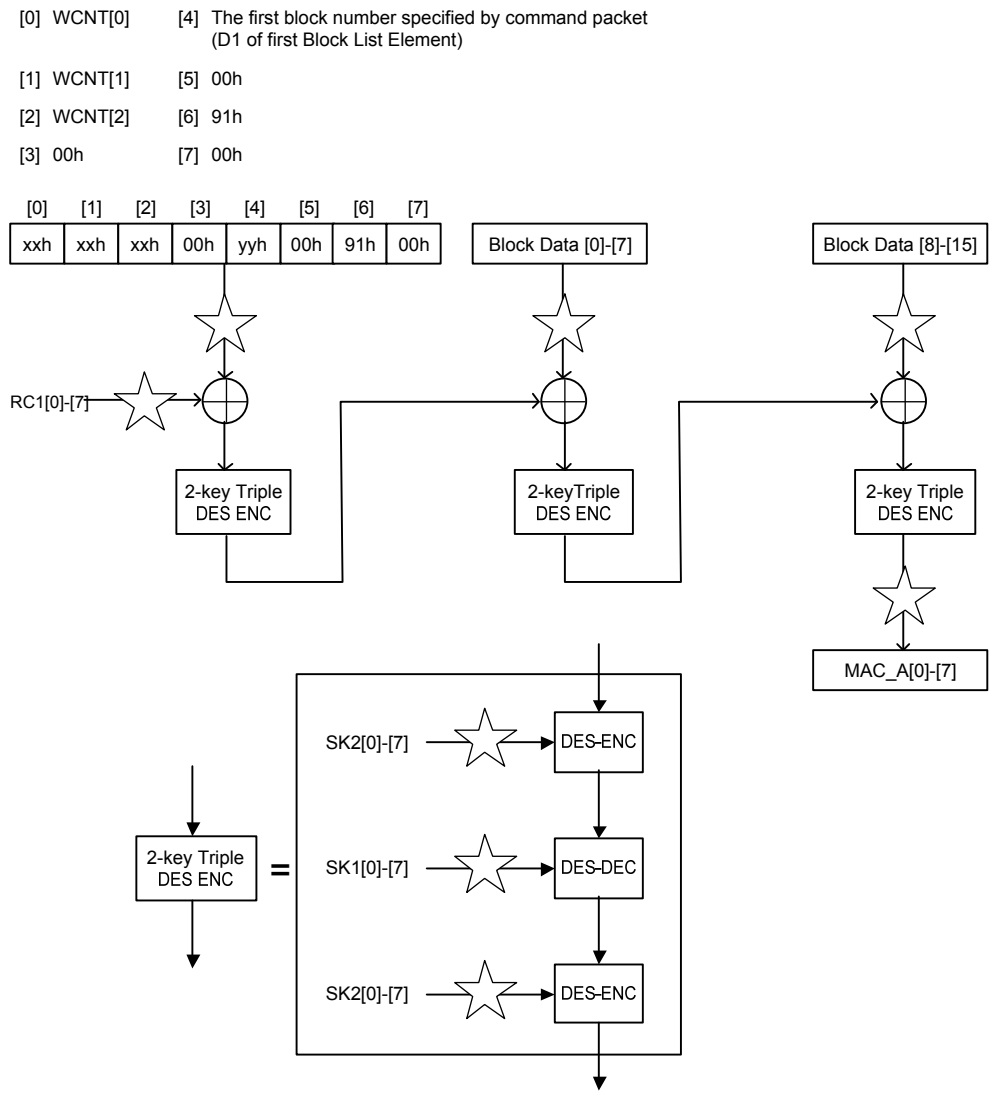



**Figure 5-4: MAC generation procedure for data-write (MAC_A Block)**

MAC is generated by a 2-key Triple DES-CBC operation using session keys (SK2, SK1) as the key, data to be written to Block as the plain text, and Random Challenge (RC1) as the initial vector. MAC is generated from the data and Block Number of Block to be written, and the WCNT value. Note that, compared with the MAC generation algorithm for data-read, the order of SK1[0]-[7] and SK2[0]-[7] is reversed.

MAC_A Block handles Write With MAC. MAC Block does not handle Write With MAC.

When MAC_A Block and any one of the S_PAD0-S_PAD13 Block, REG Block, CK Block, CKV Block, or STATE Block are written by using the Write Without Encryption command, FeliCa Lite-S generates MAC from the Block data written, Block number, and WCNT value. The generated MAC value is compared with the written MAC value of MAC_A Block, and then (only if they match) the written data is stored in non-volatile memory. If STATE Block is written, however, the data is stored in volatile memory.

# 5.3 Relationship between Block List specification and the MAC value to be read

The MAC value is generated for the data of Block read by the Read Without Encryption command by using the following rules, and they are stored in MAC Block and MAC_A Block.

- The MAC value generated by reading either MAC Block or MAC_A Block with the Read Without Encryption command is read.
- The reading of data from up to 4 Blocks is possible with a single execution of the Read Without Encryption command (In other words, generation of the MAC value is possible for up to 3 Blocks).
- MAC Block and MAC_A Block are initialized to 0 at the start of the execution of the Read Without Encryption command.
- Values read from MAC Block and MAC_A Block are not used for MAC calculation.
- MAC Block and MAC_A Block cannot be specified at the same time with a single execution of the Read Without Encryption command.
- If multiple MAC Blocks in the Block List are specified, the MAC value calculated from the entire read-data that is located before the specified MAC Block is read.
- If multiple MAC_A Blocks are selected within the Block List, only the MAC value of the MAC_A Block that is selected last is readable. The values read out of other MAC_A Blocks are ALL_00h.

Figure 5-5 to Figure 5-8 show how to specify Block List in the Read Without Encryption command, with examples of the MAC value to be read.

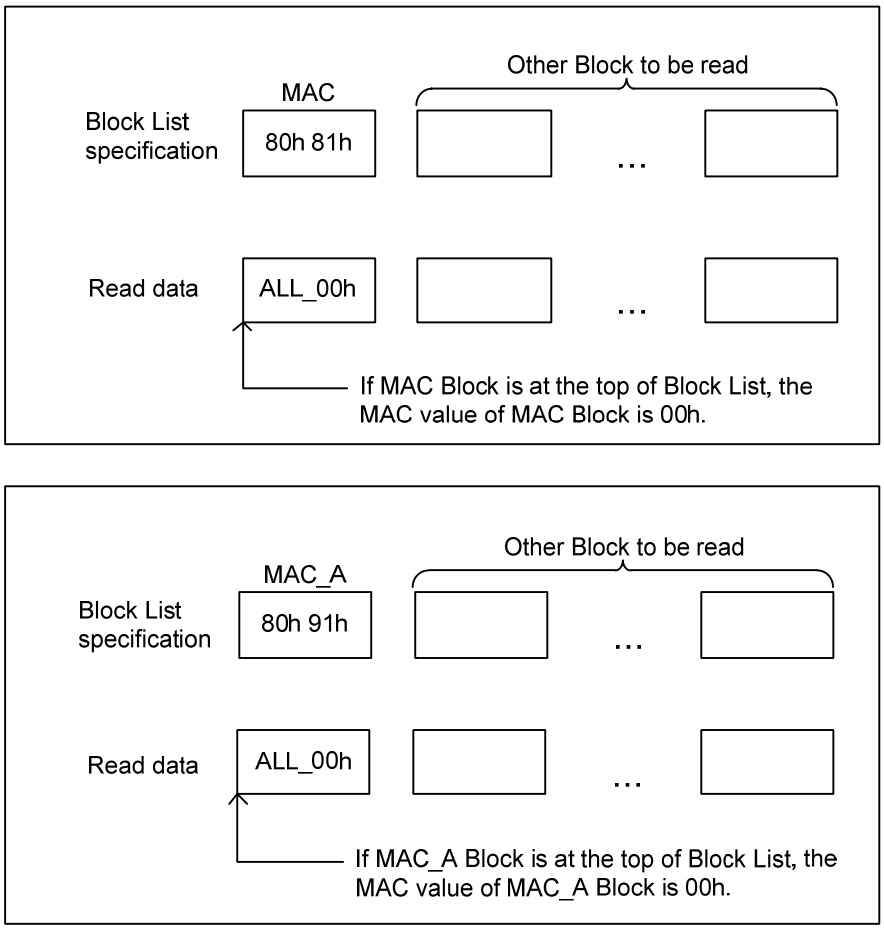



**Figure 5-5: Example 1 of generation of the MAC and MAC_A value**

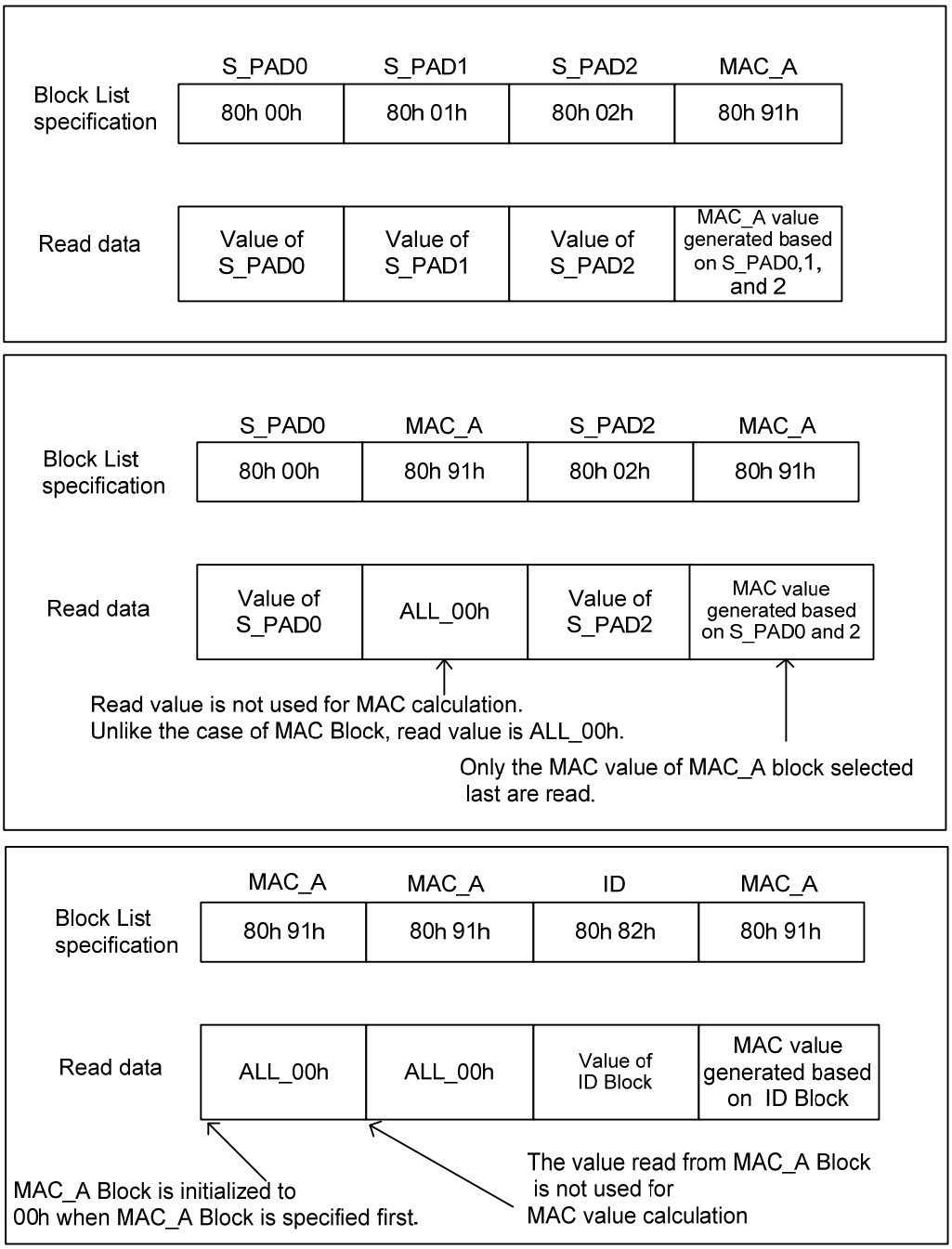



**Figure 5-6: Example 2 of generation of the MAC_A values**

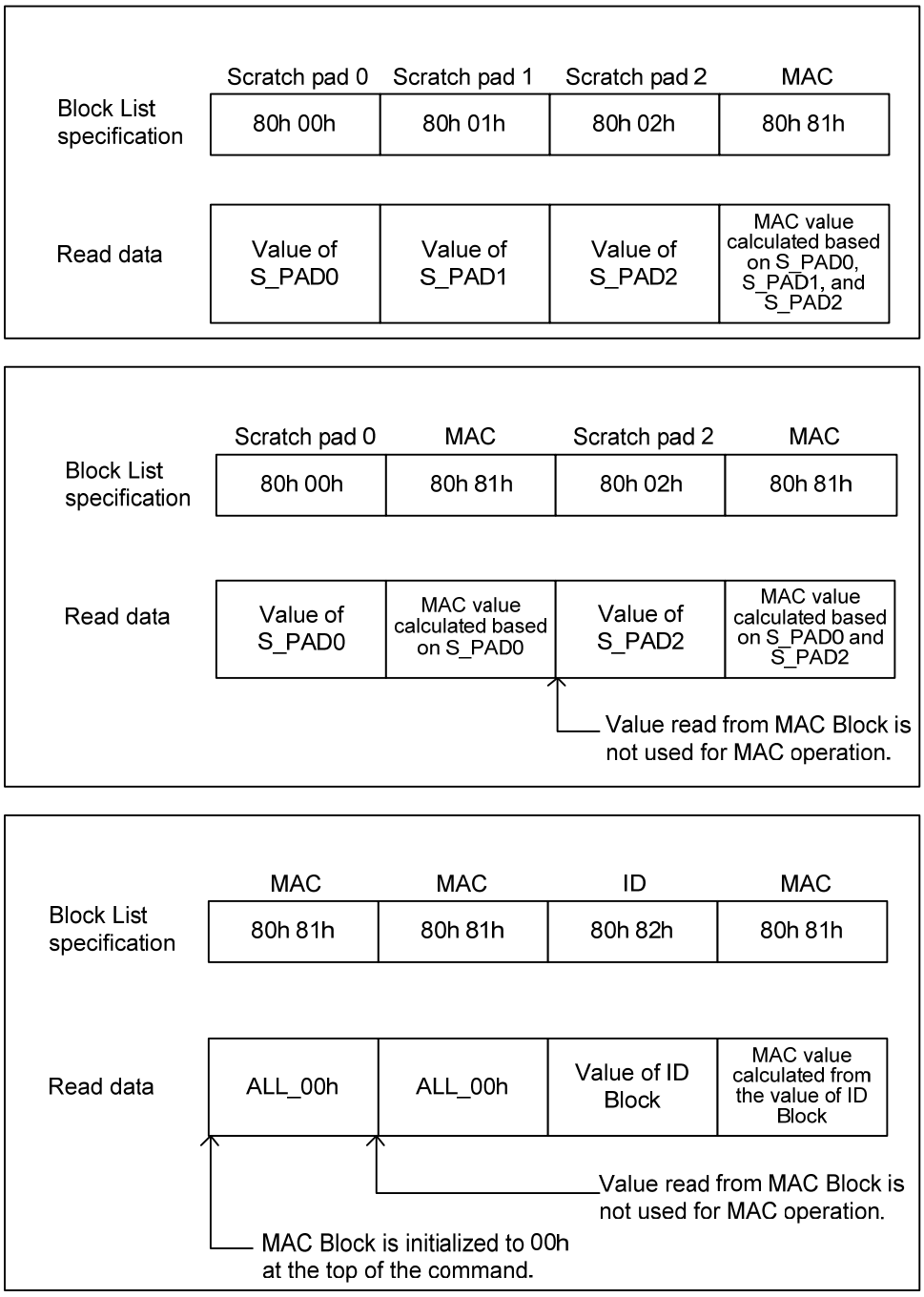



**Figure 5-7: Example 3 of generation of the MAC value**

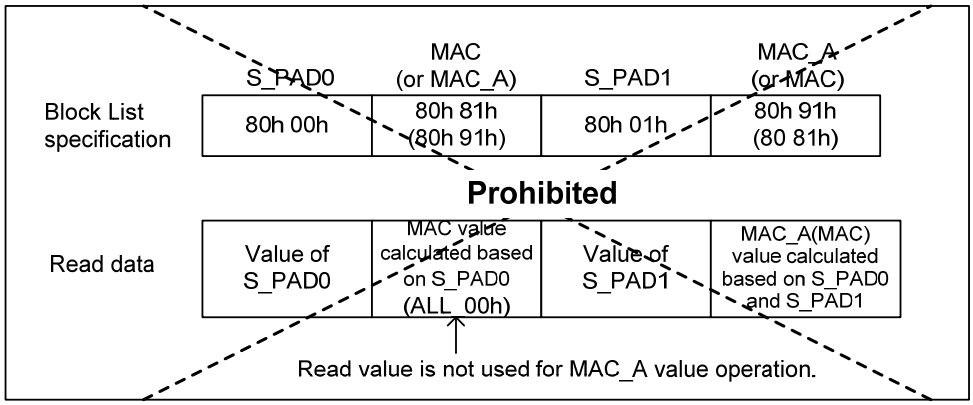



**Figure 5-8: Example 4 of generation of the MAC and MAC_A values**

# 5.4 Security protocol

## 5.4.1 Internal Authentication

Process of authentication from the Reader/Writer to a card is known as Internal Authentication. Successful authentication is determined, based on whether the card has a valid key. Figure 5-9 shows how to perform Internal Authentication using the MAC-generation function supplied with this product.

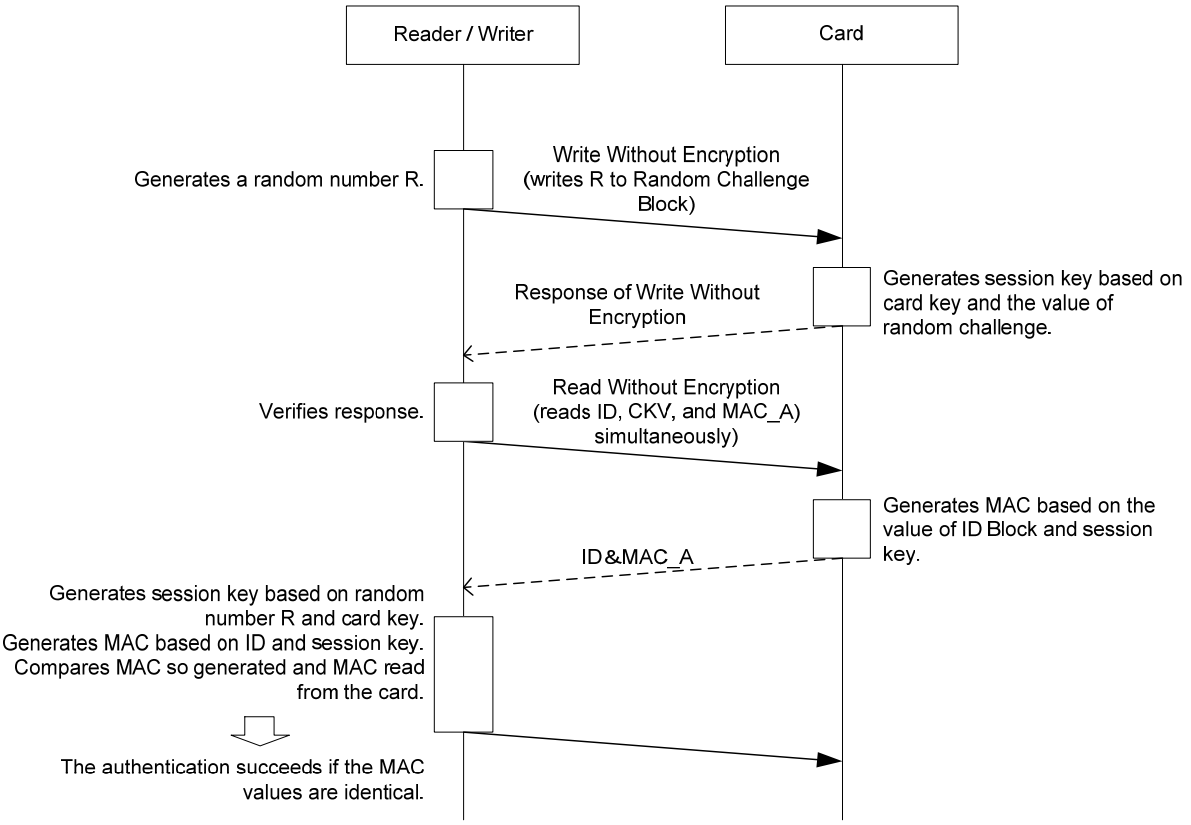



**Figure 5-9: Internal Authentication**

Both the Reader/Writer and the card calculate MAC for ID Block using their own keys. When these two MAC values are compared and if the values match, you can confirm that the Reader/Writer and the card share a common key. In this way, direct transmission and reception of the key becomes unnecessary. When generating a session key, the value written from the Reader/Writer is used. Using a random number for this value, you can prevent impersonation by a counterfeit card that reuses a response.

MAC_A Block is used for Internal Authentication. (MAC block is used for compatibility with FeliCa Lite.)

## 5.4.2 External Authentication and Mutual Authentication

The process of authentication from a card to the Reader/Writer is known as External Authentication. Successful authentication is determined, based on whether the Reader/Writer has a valid key.

In FeliCa Lite-S, Mutual Authentication can be performed by using both Internal Authentication explained in the section 5.4.1 “Internal Authentication” and External Authentication.

Figure 5-10 shows how to perform External Authentication using the MAC-generation function supplied with this product, and Mutual Authentication.

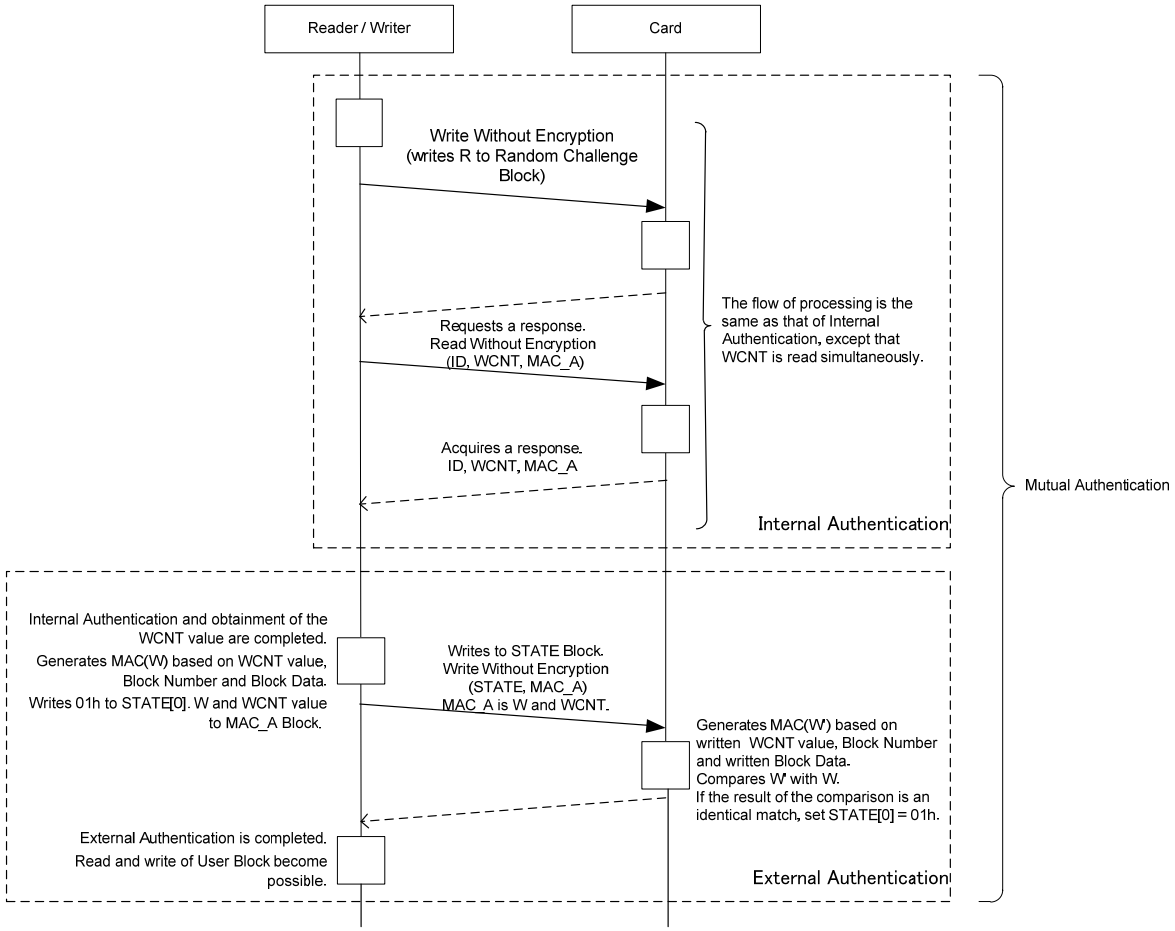



**Figure 5-10: External Authentication and Mutual Authentication**

## 5.4.3 Read With MAC

For the data stored in a card, you can prevent direct falsification of the stored data by setting RO permission for Block into which such data is stored. It is impossible, however, to prevent falsification of data on the communication path. Figure 5-11: Read With MAC
shows an example of how to detect falsification of data on the communication path with MAC.

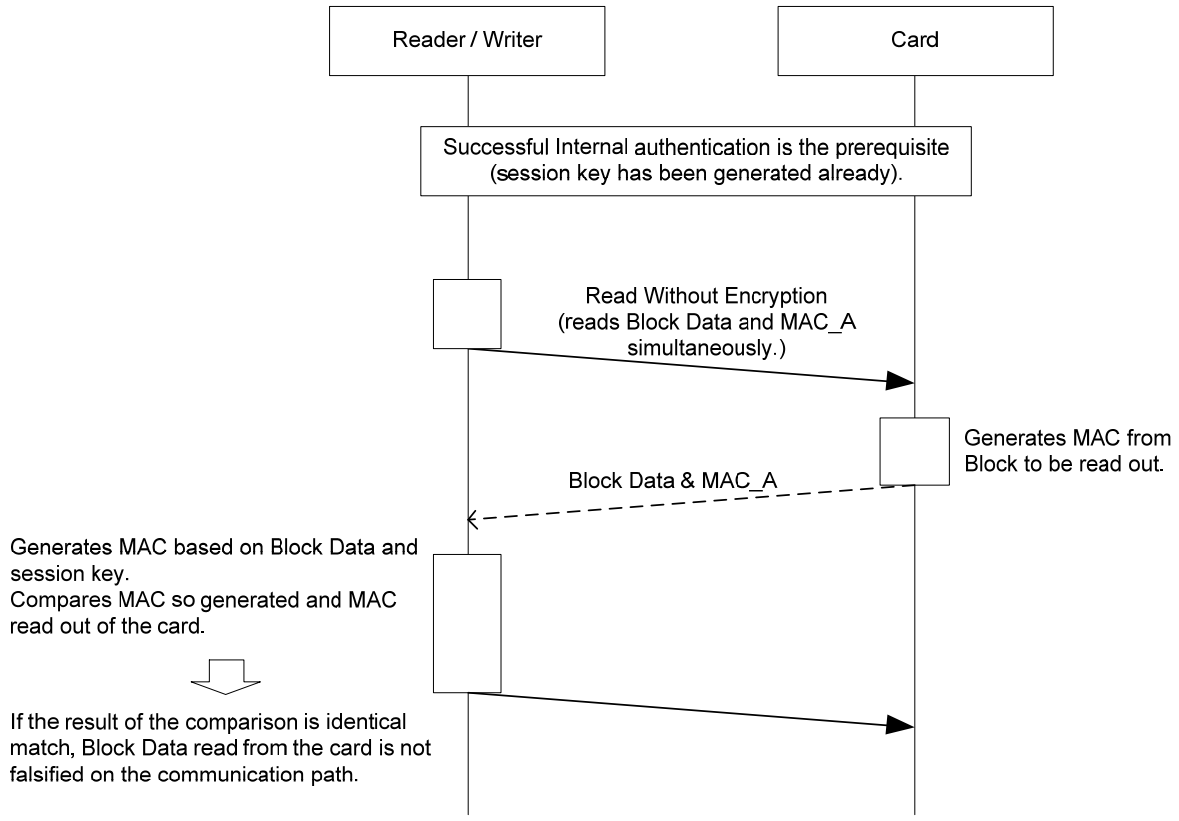



**Figure 5-11: Read With MAC**

MAC_A Block is used for Read With MAC. (MAC block is used for compatibility with FeliCa Lite.)

## 5.4.4 Write With MAC

User Block can be set to accept Write With MAC. Therefore, data falsification can be prevented and detected without a signature. Figure 5-12: Write With MAC

shows an example of how to prevent data falsification with MAC_A Block.

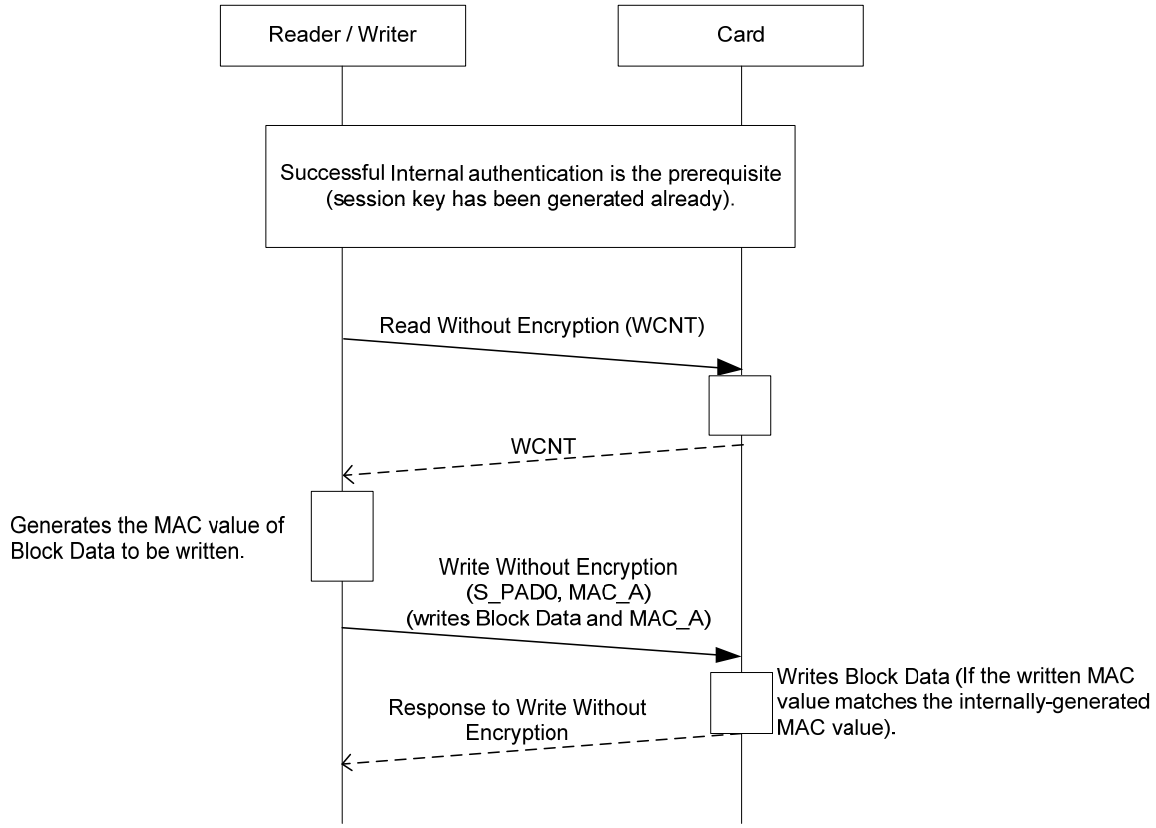



**Figure 5-12: Write With MAC**

## 5.4.5 Protected data-read from RW Permission Block

FeliCa Lite-S can detect falsification of Blocks whose access permission are set to RW in the same way as FeliCa Lite. Note that this is not necessary when Write With MAC is used.

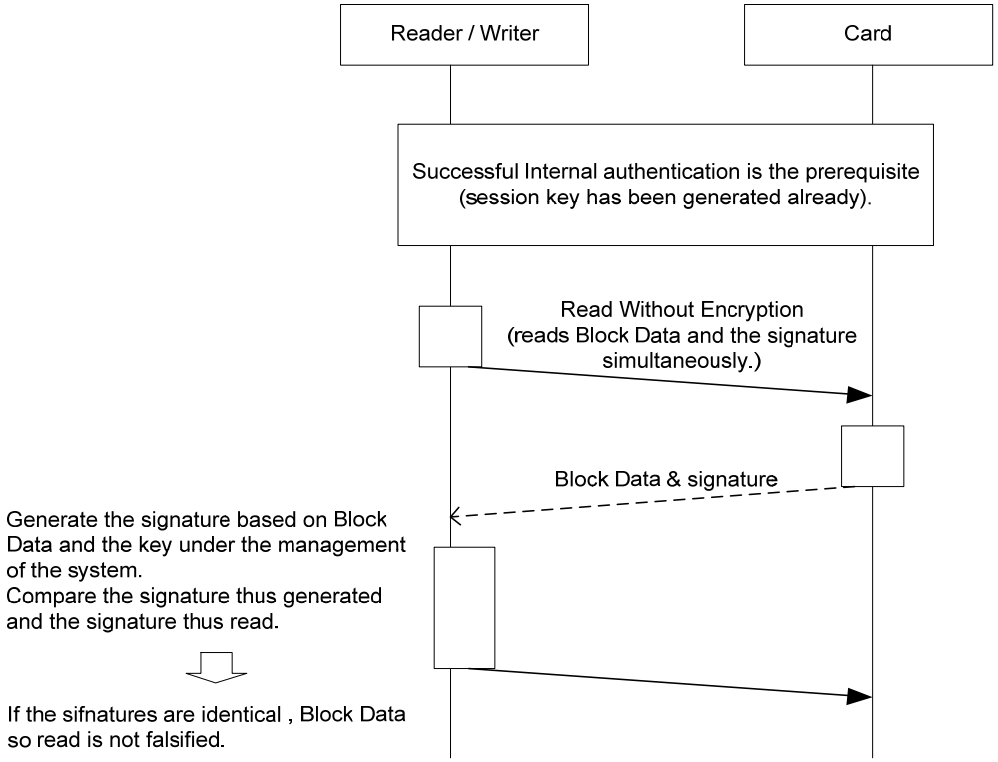



**Figure 5-13: Protected data-read to RW Block**

Store Block Data and the signature for it beforehand, and read both of them simultaneously at the time of data-read, and then compare the signature generated by the system side with the signature read from the card.

As mentioned in section 5.4.4 “Write With MAC”, when you set User Block to accept Write With MAC, data falsification can be prevented without a signature.

In this case, falsification of data on the communication channel can be detected by the method described in section 5.4.3 “Read With MAC”.

# 6 Inspection

This chapter describes the inspection items and procedures, which can be performed after implementation.

By taking your quality standards, processing capacity, and in-process defect rate over time into account, determine the inspection items/procedures to perform.

## 6.1 Applicable Reader/Writer models

You can perform the inspection using a FeliCa-compatible Reader/Writer which supports Polling, Read Without Encryption, and Write without Encryption, or NFC-compatible Reader/Writer supporting these commands.

If the Reader/Writer to be used for the inspection supports 424kbps communication, you can perform the inspection at 424kbps.

# 6.2 Inspection Items/Procedure

## 6.2.1 Inspection Items

Inspection Items of FeliCa Lite-S are listed in Table 6-1. During execution of command sequences, try to prevent the interruption of electrical power supplied to the card.

**Table 6-1: Inspection Items**

In this table: RWE = Read Without Encryption, WWE = Write Without Encryption

| No. | Inspection Items | Command | Details of inspection | Inspection Purposes |
|---|---|---|---|---|
| T1 | Verification of Polling response | Polling | Verifies that the object of inspection is FeliCa Lite-S.<br>Verifies that System Code is correct. | Detects assembly failure, retention error, defects in analogue circuits, EEPRROM, and logic circuits. |
| T2 | Verification of the result of CRC check<br>[Optional] | RWE | Verifies that the data in the non-volatile memory is correct. | Detects retention error and defects in EEPROM and logic circuits. |
| T3 | Inspection of S_PAD<br>[Optional] | WWE | Performs read-write inspection of scratch pad. | Detects defects in EEPROM and logic circuits. |
| T4 | Inspection of REG Block [Optional] | WWE | Performs read-write inspection of REG Block. | Detects defects in EEPROM |
| T5 | Inspection of ID Block, SER_C Block, CKV Block, and MC Block<br>[Optional] | WWE | Performs read-write inspection of ID Block, SER_C Block, CKV Block, and MC Block. | Detects defects in EEPROM |
| T6 | MAC generation test and inspection of CK Block [Optional] | WWE<br>RWE | Verifies correct operation of authentication function.<br>Performs read-write inspection of CK Block.*1 | Detects defects in EEPROM and logic circuits |

*1 The initial key is stored in the CK Block to prevent the operation with easily guessed key values (ex. ALL_00h). You may rewrite and perform the operation with the key value set to ALL_00h.

The Inspection Item T1 can be used to detect defected IC chips due to assembly failure and stress during the process by performing it at the end of each manufacturing process.

The Inspection Item T1 can also be used as an acceptance test when you receive inlay products (module products) of FeliCa Lite-S from other manufacturers.

### 6.2.2 Inspection Procedure

Details of the inspection procedures for each inspection items are listed in Table 6-2 through Table 6-8. In the inspection procedures described here, note the following:

- When you write data, FeliCa Lite-S stores written data to a temporary area (known as the write buffer) in non-volatile memory to perform data protection against power interruption described in section 3.6.1 “Data protection function against power interruption”. Therefore, it is assumed that inspections T3 to T6 are performed to include the write buffer.
- FeliCa Lite-S compares the CRC calculated from written data with the CRC calculated from data stored in non-volatile memory after data-write and, only if they match, returns a response indicating normal termination. Therefore, read inspection for confirming whether data-write is performed successfully is not required.
- In the data storage area in non-volatile memory, the CRC calculated from the data of each Block is also stored, as well as written data. Data pattern A is added to check 0b and 1b of all bits of the area where CRC is stored.

**Table 6-2: Inspection Procedure of “T1: Verification of Polling response”**

| No. | Procedure |
|---|---|
| 1 | Execute Polling command by specifying the System Code of 88B4h. |

**Table 6-3: Inspection Procedure of “T2: Verification of the result of CRC check [Optional]”**

| No. | Procedure |
|---|---|
| 1 | Read CRC_CHECK |

**Table 6-4: Inspection Procedure of “T3: Inspection of S_PAD [Optional]"**

| No. | Detailed procedures |
|---|---|
| 1 | Writes ALL_FFh to S_PAD0. |
| 2 | Writes ALL_FFh to S_PAD1. |
| 3 | Writes ALL_FFh to S_PAD2. |
| | : |
| 14 | Writes ALL_FFh to S_PAD13. |
| 15 | Writes ALL_FFh to S_PAD13 again. |
| 16 | Writes data pattern A to S_PAD0. |
| 17 | Writes data pattern A to S_PAD1. |
| 18 | Writes data pattern A to S_PAD2. |
| | : |
| 29 | Writes data pattern A to S_PAD13. |
| 30 | Writes data pattern A to S_PAD13 again |
| 31 | Writes ALL_00h to S_PAD0. |
| 32 | Writes ALL_00h to S_PAD1. |
| 33 | Writes ALL_00h to S_PAD2. |
| 34 | Writes ALL_00h to S_PAD3. |
| | : |
| 43 | Writes ALL_00h to S_PAD12. |
| 44 | Writes ALL_00h to S_PAD13. |
| 45 | Writes ALL_00h to S_PAD13 again. |

**Table 6-5: Inspection Procedure of “T4: Inspection of REG Block [Optional]”**

| No. | Detailed procedures |
|---|---|
| 1 | Writes ALL_FFh to REG. |
| 2 | Writes ALL_00h to S_PAD0. |
| 3 | Writes data pattern A to S_PAD0. |
| 4 | Writes data pattern A to S_PAD0 again. |
| 5 | Writes ALL_00h to S_PAD0. |
| 6 | Writes ALL_FFh to REG. |

**Table 6-6: Inspection Procedure of "T5: Inspection of ID Block, SER_C Block, CKV Block, and MC Block [Optional]"**

| No | Detailed procedures |
|---|---|
| 1 | Writes ALL_FFh to ID. |
| 2 | Writes ALL_FFh to ID again. |
| 3 | Writes ALL_00h to ID. |
| 4 | Writes ALL_00h to ID again. |
| 5 | Writes data pattern F to SER_C. |
| 6 | Writes data pattern F to SER_C again. |
| 7 | Writes ALL_00h to SER_C. |
| 8 | Writes ALL_00h to SER_C again. |
| 9 | Writes data pattern F to CKV. |
| 10 | Writes data pattern F to CKV again. |
| 11 | Writes data pattern A to CKV. |
| 12 | Writes data pattern A to CKV again. |
| 13 | Writes ALL_00h to CKV. |
| 14 | Writes ALL_00h to CKV again. |
| 14 | Writes data pattern B to MC. |
| 16 | Writes data pattern B to MC again. |
| 17 | Writes data pattern A to MC. |
| 18 | Writes data pattern A to MC again. |
| 19 | Writes data pattern C to MC. |
| 20 | Writes data pattern C to MC again. |

### Table 6-7: Inspection Procedure of “T6: MAC generation test and inspection of CK Block [Optional]”

| No | Detailed procedures |
|---|---|
| 1 | Writes ALL_FFh to CK. |
| 2 | Writes ALL_FFh to CK again. |
| 3 | Writes data pattern D to ID Block. |
| 4 | Writes data pattern E to RC Block. |
| 5 | Reads ID / MAC_A. |
| 6 | Reads ID / ID / MAC_A. |
| 7 | Reads ID / ID / ID / MAC_A. |
| 8 | Writes data pattern A to CK. |
| 9 | Writes data pattern A to CK again. |
| 10 | Writes ALL_00h to CK. |
| 11 | Writes ALL_00h to ID. |

### Table 6-8: Data pattern

| Data pattern | Block | Block Data(HEX) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | S_PAD0-13, CK | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 84CF |
| | CKV | 84CF | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 |
| | MC | 0000 | 0000 | 0700 | 0000 | 0000 | 00C3 | 7A00 | 0000 |
| B | MC | 0000 | 00FF | 07FF | FFFF | FFFF | FFFF | FF00 | 0000 |
| C | MC | FFFF | FF00 | 0700 | 0000 | 0000 | 0000 | 0000 | 0000 |
| D | ID | 299F | FA53 | AB75 | 876E | 574E | 102A | 9416 | BC8E |
| E | RC | F187 | 5A01 | F9B2 | 9E4C | 06A1 | CEC4 | 1655 | 85CF |
| F | SER_C, CKV | FFFF | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 | 0000 |

# 6.3 Examples of Command/Response Packet Data

This section describes examples of Command/Response Packet Data (where the unit of size is in Bytes) to be executed in each step of the inspection procedure. For the procedure to execute each command from the Reader/Writer, see the manual supplied with the Reader/Writer you are using.

**<Caution>**

Inspection procedure described in this section is just one example.
The example to be described in this section omits some procedures related to the area where CRC is stored (see “3” in section 6.2.2 “Inspection Procedure”).

## 6.3.1 T1: Verification of Polling response

During capture of the card, simultaneously acquire IDm, which is necessary for subsequent commands. Also check and verify that the value of System Code is correct.

**<Items to be verified>**

Execute Command Packet Data, and check and verify Response Packet Data.

**Command Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 00 | Polling |
| System Code | 2 | 88 B4 | |
| Request Code | 1 | 00 | |
| Time Slot | 1 | 00 | |

**Response Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 01 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | It is unnecessary to verify xx at the time of inspection. You must, however, acquire the value to be able to specify IDm in subsequent commands. |
| PMm | 8 | YY YY 00 00 00 YY YY 00 | It is unnecessary to verify YY at the time of inspection. |

## 6.3.2 T2: Verification of the result of CRC check [Optional]

Perform to see whether the data in the non-volatile memory is correct.

**<Items to be verified>**

Execute Command Packet Data, and check and verify Response Packet Data.

**Command Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 06 | Read Without Encryption |
| IDm | 8 | | Specify IDm described in the procedure of section 6.3.1 “T1: Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 0B 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 A0 | Specify CRC_CHECK Block. |

**Response Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 07 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |
| Number of Block | 1 | 01 | |
| Block Data | 16 | 00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 | |

### 6.3.3 T3: Inspection of S_PAD [Optional]

Perform read-write inspection for S_PAD and write buffer.

**<Items to be verified>**

Execute Command Packet Data, and check and verify Response Packet Data.

**Command Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify IDm acquired in the procedure of section 6.3.1 “T1: Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 XX | For the value of XX, see “<Command Execution Procedure>” on the next page. |
| Block Data | 16 | YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY | For the value of YY, see “<Command Execution Procedure>” on the next page. |

**Response Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

**<Command Execution Procedure>**

While changing XX and YY, execute Command Packet Data. Table 7-1 shows the order of execution of the values of XX and YY.

**Table 6-9: Order of execution of the values of XX and YY**

| Order | Value of XX | Value of YY | Order | Value of XX | Value of YY |
|---|---|---|---|---|---|
| 1 | 00h | FFh | 16 | 00h | 00h |
| 2 | 01h | | 17 | 01h | |
| 3 | 02h | | 18 | 02h | |
| 4 | 03h | | 19 | 03h | |
| 5 | 04h | | 20 | 04h | |
| 6 | 05h | | 21 | 05h | |
| 7 | 06h | | 22 | 06h | |
| 8 | 07h | | 23 | 07h | |
| 9 | 08h | | 24 | 08h | |
| 10 | 09h | | 25 | 09h | |
| 11 | 0Ah | | 26 | 0Ah | |
| 12 | 0Bh | | 27 | 0Bh | |
| 13 | 0Ch | | 28 | 0Ch | |
| 14 | 0Dh | | 29 | 0Dh | |
| 15 | 0Dh | | 30 | 0Dh | |

### 6.3.4 T4: Inspection of REG Block [Optional]

Perform read-write inspection for REG Block.

#### <Items to be verified>

Execute Command Packet Data, and check and verify Response Packet Data.

#### Command Packet Data

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 6.3.1"T1: Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 XX | For the value of XX, see "<Command Execution Procedure>" below. |
| Block Data | 16 | YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY YY | For the value of YY, see "<Command Execution Procedure>" below. |

#### Response Packet Data

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

#### <Command execution procedure>

While changing XX and YY, execute Command Packet Data. Table 6-10: Order of execution of the values of XX and YY shows the order of execution of the values of XX and YY.

**Table 6-10: Order of execution of the values of XX and YY**

| Order | Value of XX | Value of YY |
|---|---|---|
| 1 | 0Eh | FFh |
| 2 | 00h | 00h |
| 3 | 00h | 00h |
| 4 | 0Eh | FFh |

## 6.3.5 T5: Inspection of ID, SER_C, CKV, and MC Blocks [Optional]

Perform read-write inspection to ID Block, SER_C Block, CKV Block, and MC Block.

**<Items to be verified>**

Execute Command Packet Data, and check and verify Response Packet Data.

**Command Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 6.3.1“T1: Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | W1 W2 | For the values of W1 and W2, see “<Command Execution Procedure>” on the next page. |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 XX | For the value of XX, see “<Command Execution Procedure>” on the next page. |
| Block Data | 16 | YY YY ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ | For the values of YY and ZZ, see “<Command Execution Procedure>” on the next page. |

**Response Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

### <Command execution procedure>

While changing W1, W2, XX, YY, and ZZ, execute Command Packet Data. Table 6-11: Order of execution of the values of W1, W2, XX, YY, and ZZ shows the order of execution of the values of W1, W2, XX, YY, and ZZ.

**Table 6-11: Order of execution of the values of W1, W2, XX, YY, and ZZ**

| Order | Value of W1 | Value of W2 | Value of XX | Value of YY | Value of ZZ |
|---|---|---|---|---|---|
| 1 | 09h | 00h | 82h | FFh | FFh |
| 2 | | | | | |
| 3 | | | | 00h | 00h |
| 4 | | | | | |
| 5 | | | 84h | FFh | 00h |
| 6 | C9h | FFh | | | |
| 7 | | | | 00h | |
| 8 | 09h | 00h | | | |
| 9 | | | 86h | FFh | 00h |
| 10 | | | | | |
| 11 | | | | 00h | |
| 12 | | | | | |
| 13 | | | 88h | *1 | |
| 14 | | | | | |
| 15 | | | | *2 | |
| 16 | | | | | |

*1 0000 00FF 07FF FFFF FFFF FFFF FF00 0000

*2 FFFF FF00 0700 0000 0000 0000 0000 0000

## 6.3.6 T6: MAC generation test and inspection of CK Block [Optional]

Perform the test for the MAC-generation function. At the same time, perform read-write inspection for CK Block.

**<Items to be verified>**

Execute Command Packet Data, and check and verify that Response Packet Data1 and 2 match.

**Command Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 6.3.1“T1: Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | | See “<Command Execution Procedure>” on the next page. |
| Block Data | 16 | | See “<Command Execution Procedure>” on the next page. |

**Response Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

### Command Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 06 | Read Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 6.3.1“T1: Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 0B 00 | |
| Number of Block | 1 | | See “<Command Execution Procedure>” on the next page. |
| Block List | | | See “<Command Execution Procedure>” on the next page. |

### Response Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 07 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |
| Number of Block | 1 | | See “<Command Execution Procedure>” on the next page. |
| Block Data | | | See “<Command Execution Procedure>” on the next page. |

### <Command Execution Procedure>

While changing Number of Block, Block List, and Block Data, execute Command Packet Data 1 and Command Packet Data 2. Table 6-12 shows the order of execution of Command Packet Data and the values of the arguments.

**Table 6-12: Execution procedure of MAC generation test and inspection of CK Block**

| Order | Command Packet Data | Number of Block | Block List(HEX) | Block Data(HEX) |
|---|---|---|---|---|
| 1 | 1 | – | 80 87 | FF FF FF FF FF FF FF FF<br>FF FF FF FF FF FF FF FF |
| 2 | 1 | – | 80 87 | FF FF FF FF FF FF FF FF<br>FF FF FF FF FF FF FF FF |
| 3 | 1 | – | 80 82 | 29 9F FA 53 AB 75 87 6E<br>57 4E 10 2A 94 16 BC 8E |
| 4 | 1 | – | 80 80 | F1 87 5A 01 F9 B2 9E 4C<br>06 A1 CE C4 16 55 85 CF |
| 5 | 2 | 02h | 80 82 80 91 | 29 9F FA 53 AB 75 87 6E<br>57 4E 10 2A 94 16 BC 8E<br>EE F4 B0 BB 5E 3B 6C 8B<br>00 00 00 00 00 00 00 00 |
| 6 | 2 | 03h | 80 82 80 82 80 91 | 29 9F FA 53 AB 75 87 6E<br>57 4E 10 2A 94 16 BC 8E<br>29 9F FA 53 AB 75 87 6E<br>57 4E 10 2A 94 16 BC 8E<br>4E C7 C5 5A 17 29 CA AE<br>00 00 00 00 00 00 00 00 |
| 7 | 2 | 04h | 80 82 80 82 80 82 80 91 | 29 9F FA 53 AB 75 87 6E<br>57 4E 10 2A 94 16 BC 8E<br>29 9F FA 53 AB 75 87 6E<br>57 4E 10 2A 94 16 BC 8E<br>29 9F FA 53 AB 75 87 6E<br>57 4E 10 2A 94 16 BC 8E<br>D9 9A E9 6E 0C 48 2C E4<br>00 00 00 00 00 00 00 00 |
| 8 | 1 | – | 80 87 | 00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 |
| 9 | 1 | – | 80 82 | 00 00 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 |

# 7 Issuance

The process for writing various types of information essential for performing the operations of the service is known as “Issuance”. This chapter defines the procedure to perform the process of issuance for FeliCa Lite-S.

## 7.1 Overview of issuance procedure

Issuance of FeliCa Lite-S is divided into three steps, i.e., the 0th issuance, the 1st issuance and the 2nd issuance. The 0th issuance is the step to write three types of information to the IC chip: i.e., IDd (IDm), PMm and System Code. The 0th issuance is completed when the IC is shipped from the factory, so you do not need to be aware of it. Table 7-1: Issuance steps and information to be set to the rewrite prevention state shows the purposes of the 1st and the 2nd issuance, as well as the data to be written to the IC.

**Table 7-1: Issuance steps and information to be set to the rewrite prevention state**

| | 1st issuance | 2nd issuance |
|---|---|---|
| Purpose | To set up:<br>- System Block to rewrite prevention state<br>- access permissions of User Block, CK Block, CKV Block, and STATE Block as necessary<br>- User Block as necessary | To set up access permissions of User Block and STATE Block and set the access permissions to be unchangeable thereafter. |
| Information to be fixed | DFC<br>Service Code<br>ID<br>(Card Key)*1<br>(Card Key Version)*2<br>NDEF option<br>A part of User Block | User Block |
| Execution timing | After manufacture of card | ← |
| Required device | FeliCa Reader/Writer, NFC Reader/Writer | ← |
| Performer | Card issuer | ← |

*1, *2Card Key and Card Key Version are fixed when MC_CKCKV_W_MAC_A = 00h.

The 1st and the 2nd issuance are divided, for convenience, to show that the process of issuance can be divided on a data-by-data basis to be written. There is no problem, however, even if both the 1st and the 2nd issuance procedures are performed simultaneously.

# 7.2 Applicable Reader/Writer models

Issuance of FeliCa Lite-S is performed using the following commands: Polling, Read Without Encryption, and Write Without Encryption. Issuance can be performed using a FeliCa-compatible or NFC-compatible Reader/Writer that supports these commands.

# 7.3 1st issuance procedure

The purpose of the 1st issuance is to set the rewrite prevention setting of System Block, to set the rewrite prevention setting of CK Block, CKV Block, and STATE Block as necessary, and to set NDEF-compatible option.

You can also perform the setup of User Block as necessary.

If you need to setup FeliCa Lite-S as the same as FeliCa Lite, you can issue FeliCa Lite-S by performing the same issuance procedure as FeliCa Lite as described in “FeliCa Lite User’s Manual”.

Table 7-2: List of 1st issuance procedures shows the list of the 1st issuance procedures.

**Table 7-2: List of 1st issuance procedures**

| | Procedure | Command | Content of procedure |
|---|---|---|---|
| 1 | Verification of Polling response | Polling | Verify that object of issuance is FeliCa Lite-S.<br>Verify that System Code has correct value. |
| 2 | Setup of ID | Read Without Encryption<br>Write Without Encryption | Set ID. |
| 3 | Writing of Card Key | | Set Card Key. |
| 4 | Verification of Card Key | | Verify that Card Key is set correctly. |
| 5 | Writing of Card Key Version | | Set Card Key Version. |
| 6 | Writing of User Block (optional) | | Set values to User Block. |
| 7 | Setting of rewrite prevention for System Block/access permission for each Block/NDEF-compatible option | | Set RO permission for System Block.<br>Also set the access permission of each Block and NDEF-compatible option. |
| 8 | Committing of issuance | – | Shut the electrical power off and fix the content of the 1st issuance. |

Packet Data of commands to be executed in each issuance procedures are described in the following sections (where the unit of size is Byte).

## 7.3.1 Verification of Polling response

Execute the content of section 6.3.1“T1: Verification of Polling response”.

## 7.3.2 Setup of ID

Write ID, and verify that the ID was correctly written.
Store ALL_00h in ID[0]-[7], and arbitrary values in ID[10]-[15].
Store DFC (if DFC is not used, store 0000h) in ID[8][9].

**<Items to be verified>**

Execute Command Packet Data 1, Command Packet Data 2 in the order shown below. Check and verify each Response Packet Data.

**Command Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1"Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 82 | |
| Block Data | 16 | | Specify the value of ID to set. |

**Response Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

### Command Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 06 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1"Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 0B 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 82 | |

### Response Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 07 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |
| Number of Block | 1 | 01 | |
| Block Data | 16 | | Verify the value is the same as the written ID value[*1] |

[*1] Note that the value read from ID[0]-[7] becomes IDd instead of the value that was written in after power off. (See section 3.1.6"ID" for details)

## 7.3.3 Writing of Card Key

Write Card Key.

**<Items to be verified>**

Execute Command Packet Data, and verify Response Packet Data.

**Command Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1"Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 87 | |
| Block Data | 16 | | Specify Card Key to be set. |

**Response Packet Data**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

## 7.3.4 Verification of Card Key

Verify that Card Key is correctly written.

**<Items to be verified>**

Execute Command Packet Data 1 and Command Packet Data 2 in the order shown here, and then verify each Response Packet Data.

**Command Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1"Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 80 | |
| Block Data | 16 | | Specify arbitrary 16-Byte data. |

**Response Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

### Command Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 06 | Read Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1"Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 0B 00 | |
| Number of Block | 1 | 02 | |
| Block List | 4 | 80 82 80 91 | |

### Response Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 07 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |
| Number of Block | 1 | 02 | |
| Block Data | 32 | | ID set in this way (16 Bytes) and MAC value |

Card Key is impossible to read from the card by reading the value from CK Block, so use the MAC value that was generated based on the value of Card Key to verify that the correct data was written. Use Command Packet Data 1 to generate the session key by writing to RC Block. Next, acquire the MAC for ID by using Command Packet Data 2.

Based on ID, Card Key and the value written to RC Block, calculate the expected value of MAC read from MAC_A Block. For details of how to generate the MAC, see section 5.2.1 "MAC generation procedure for data-read". You can use MAC Block instead of MAC_A Block.

## 7.3.5 Writing of Card Key Version

Write Card Key Version, and verify that Card Key Version is correctly written.

**<Items to be verified>**

Execute Command Packet Data 1 and Command Packet Data 2 in the order shown here, and then verify each Response Packet Data.

### Command Packet Data 1

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1"Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 86 | |
| Block Data | 16 | YY YY 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 | To YY, specify Key Version to be set. |

### Response Packet Data 1

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

### Command Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 06 | Read Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1"Verification of Polling response". |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 0B 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 86 | |

### Response Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 07 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |
| Number of Block | 1 | 01 | |
| Block Data | 16 | YY YY 00 00 00 00 00 00<br>00 00 00 00 00 00 00 00 | Verify that it is the value of Card Key Version just written. |

## 7.3.6 Writing of User Block (optional)

If there is any User Block whose value needs to be set at the time of 1st Issuance, write the appropriate data in this step of the procedure.

Referring to section 7.3.2"Setup of ID" execute the command packet data that specifies the Block Data and the Block Number of the User Block that you want to set up.

## 7.3.7 Setup of rewrite prevention for System Block/access permission for each Block/NDEF-compatible option

Set System Block to the rewrite prevention state (RO permission). Also set up the NDEF-compatible option and the access permission of User Block, CK Block, CKV Block, and STATE Block.

**<Items to be verified>**

Execute Command Packet Data 1 and Command Packet Data 2 in the order shown here, and then verify each Response Packet Data.

**Command Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1“Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 88 | |
| Block Data | 16 | X1 X2 00 YY 07 ZZ X3 X4 X5 X6 X7 X8 X9 00 00 00 | For the values of X1 to X9, YY, and ZZ, see on the next page. |

**Response Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

### Command Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 06 | Read Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.3.1“Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 0B 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 88 | |

### Response Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 07 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |
| Number of Block | 1 | 01 | |
| Block Data | 16 | X1 X2 00 YY 07 ZZ X3 X4<br>X5 X6 X7 X8 X9 00 00 00 | |

Values of X1 to X9 are determined depending on Block for which the rewrite prevention state was set at the completion of the 1st issuance. Set 1b in the highest bit of X2, however. For the procedure to specify X1 to X9, see section 3.1.12“MC”.

Specify 01h to YY when compatibility with NDEF is necessary. In other cases, set 00h.

Specify 01h to ZZ if you rewrite CK Block and CKV Block with MAC after the 1st issuance or, if not, specify 00h to ZZ.

## 7.3.8 Committing of issuance

The 1st issuance is committed by shutting off the electrical power to the IC. By switching the RF output of the Reader/Writer off, you can shut off the electrical power to the IC chip. For the procedure to switch off the RF output of the Reader/Writer, see the manual supplied with the Reader/Writer to be used.

# 7.4 2nd issuance procedure

The purpose of the 2nd issuance is to set up the User Block and fix the access permission of User Block and STATE Block.

**Table 7-3: List of 2nd issuance procedures**

| | Procedure | Command | Content of procedure |
|---|---|---|---|
| 1 | Verification of Polling response | Polling | Verify that the object of issuance is FeliCa Lite-S.<br>Verify that the value of System Code is correct. |
| 2 | Writing of User Block (optional) | Read Without Encryption<br>Write Without Encryption | Set values to User Block. |
| 3 | Setup of access permission for User Block and STATE Block | | Set rewrite prevention to access permission of User Block and STATE Block, and make them unchangeable thereafter. |
| 4 | Committing of issuance | – | Shut the power off to fix the content of the 2nd issuance. |

The following sections show Packet Data of commands to be executed in each procedure (the unit of size is Byte).

## 7.4.1 Verification of Polling response

Execute the content of section 6.3.1“T1: Verification of Polling response”.

## 7.4.2 Writing of User Block (optional)

For User Block which is to be set to the rewrite prevention state after writing data in the 2nd issuance, write the appropriate data in this step of the procedure.

By referring to section 7.4.3“Setup of access permission for User Block and STATE Block” execute the command packet data that specifies the Block Number and the Block Data of the User Block that you want to set up, and verify that each response packet data is as expected.

## 7.4.3 Setup of access permission for User Block and STATE Block

Set the access permission for each User Block and STATE Block, and set the access permission settings to the rewrite prevention state.

**<Items to be verified>**

Execute Command Packet Data 1 and Command Packet Data 2 in the order shown here, and then verify each Response Packet Data.

**Command Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 08 | Write Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.4.1“Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 09 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 88 | |
| Block Data | 16 | X1 X2 00 00 00 00 X3 X4 X5 X6 X7 X8 X9 00 00 00 | For the values of X1, X2…X9, see on the next page. |

**Response Packet Data 1**

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 09 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |

### Command Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Command Code | 1 | 06 | Read Without Encryption |
| IDm | 8 | | Specify the IDm acquired in the procedure of section 7.4.1“Verification of Polling response”. |
| Number of Service | 1 | 01 | |
| Service Code List | 2 | 0B 00 | |
| Number of Block | 1 | 01 | |
| Block List | 2 | 80 88 | |

### Response Packet Data 2

| | Size | Data (HEX) | Note |
|---|---|---|---|
| Response Code | 1 | 07 | |
| IDm | 8 | xx xx xx xx xx xx xx xx | IDm specified in Command Packet Data |
| Status Flag1 | 1 | 00 | |
| Status Flag2 | 1 | 00 | |
| Number of Block | 1 | 01 | |
| Block Data | 16 | xx xx 00 xx xx xx xx xx<br>xx xx xx xx xx 00 00 00 | Verify the values. |

Set the values of X1...X9 depending on the access permission of User Block to be set during the 2nd issuance. Set 0b to the highest bit of X2, however. It is impossible to write 1b to the bits of X1 and X2 to which 0b was written in the 1st issuance. It is also impossible to write 0b to the bits of X3…X9 to which 1b was written in the 1st issuance. For details of the procedure to specify X1…X9, see section 3.1.12 3.1.12“MC”.

## 7.4.4 Committing of issuance

The 2nd issuance is committed by shutting off the electrical power to the IC. By switching off the RF output of the Reader/Writer, you can shut off the electrical power to the IC chip. For the procedure to switch off the RF output of the Reader/Writer, see the manual supplied with the Reader/Writer to be used.

# Appendix A Data Format Code (DFC)

DFC is a 2-Byte value that specifies the data format stored in a card. For detailed information about DFC, see “FeliCa Technology Code Descriptions”, which you can find at the following website:

http://www.sony.net/Products/felica/business/tech-support/index.html

**<How to read the DFC from FeliCa Lite-S>**

To be able to read the DFC in the operational environment, write the DFC value in ID Block (ID[8][9]) during the 1st issuance.

**<How to apply the DFC>**

For details of the registration procedure, please contact your FeliCa Lite-S suppliers.

DFC is reserved and granted by Sony.

# Appendix B FeliCa terminology

This appendix defines the FeliCa-specific abbreviations and terms used in this document.

## B.1 Abbreviations

| | |
|---|---|
| IDd | Device ID |
| IDm | Manufacture ID |
| PMm | Manufacture Parameter |
| SF | Status Flag |
| SF1 | Status Flag1 |
| SF2 | Status Flag2 |

## B.2 Glossary

**<A>**

**Access Mode**

A value specified in Block List Element.

This value identifies the method of access to use when accessing Block Data.

**<B>**

**Big Endian**

The method to sequentially record or transfer numerical data longer than 2 Bytes, which is divided on a Byte-by-Byte basis, from the highest (i.e., most significant) Byte first.

**Block**

The minimum unit of data written to or read from memory.

**Block Data**

1. Data to be written to or read from Block.
2. Data to be stored in Block.

**Block List**

The enumeration (i.e., the ordered array) of all Block List Element instances.

**Block List Element**

Data that specifies which Service and Block Number to access.

**Block Number**

A value specified in Block List Element. This value identifies the logical location of Block Data.

**<E>**

**External Authentication**

The process of authentication performed between a card and the Reader/Writer, to verify that the Reader/Writer has the correct key.

**<I>**

**IC Code**

The 2-Byte code that uniquely identifies each type of integrated chip (IC). IC Code comprises ROM Type (1 Byte) and IC Type (1 Byte).

**IC Type**

The 1-Byte code that uniquely identifies each hardware type.

**Internal Authentication**

The process of authentication performed between a card and the Reader/Writer, to verify that the card has the correct key.

**<K>**

**Key Version**

The value that identifies each key version.

**<L>**

**Little Endian**

The method to sequentially record or transfer numerical data longer than 2 Bytes, which is divided on a Byte-by-Byte basis, from the lowest (i.e., least significant) Byte first.

**<M>**

**Manufacture ID (IDm)**

The value that comprises Manufacturer Code and Card Identification Number. The Reader/Writer uses this value to identify each card with which to communicate.

**Manufacture Parameter (PMm)**

Card-specific information that is set by the card manufacturer.

**Manufacturer Code**

The upper 2 Bytes of Manufacture ID (IDm). This value identifies the manufacturer that assigned Manufacture ID (IDm) to the card.

**<N>**

**No Response**

The operation that terminates communications without sending a response to the received command.

**<P>**

**Packet Data**

The data between the Packet Data Length field and the CRC field.

**Packet Data Length (LEN)**

The sum of the length of the Packet Data field plus the length of the Packet Data Length field (LEN)

**<R>**

**ROM Type**

The 1-Byte code that uniquely identifies the software (ROM) type of the same IC Type.

**Random Service**

Service that enables read operations or write operations by specifying Block.

**Read After Authentication**

The read operation after the successful External Authentication.

**Read With MAC**

The read operation that reads the MAC value from MAC Block or MAC_A Block simultaneously during data-reading. The Reader/Writer can verify the integrity of the read data using the MAC value.

**<S>**

**Service**

The concept that identifies both the method of access to Block Data and a set of Block Data.

**Service Attribute**

The lower 6 bits of Service Code, which determine how to access Block Data.

**Service Code**

The value that uniquely identifies each Service.

**Service Code List Order**

A value specified in Block List Element. This value specifies the target Service to access using an enumeration from Service Code List.

**Service Number**

The value in Service Code, excluding the bits that define Service Attribute.

**Simple Read**

The write operation other than Write After Authentication and Write With MAC.

**Simple Write**

The read operation other than Read After Authentication and Read With MAC.

**Status Flag**

The information that indicates the error status of a card, consisting of Status Flag1 and Status Flag2.

**System**

The logically-formatted domain that contains the FeliCa file management structure.

**System Code**

The value that uniquely identifies each System. System Code is assigned per service provider and per application. System Code of FeliCa Lite-S is 88B4h.

**System Separation**

The operation that both logically divides the memory located on a card and creates multiple logical card functions (i.e., more than one System) on that card.

**<W>**

**Write After Authentication**

The write operation after the successful External Authentication.

**Write With MAC**

The write operation that writes the MAC value to MAC_A Block simultaneously during data-writing. This write operation succeeds only when the MAC value is correct.

FeliCa Lite-S

# FeliCa Lite-S User's Manual Version 1.21


| | | |
|---|---|---|
| March 2012 | First Edition | FeliCa Business Division |
| April 2017 | Revision | |



Sony Imaging Products & Solutions Inc.

No. M741-E01-21





Printed in Japan