SONY

FeliCa

Technical Information

FeliCa Card
Technical Note
for
Software Development

Version 1.01

No. M606-E01-01

Introduction

This document summarises important information about the development of software (firmware) for contactless IC cards that operate with FeliCa technology, in terms of maintaining compatibility with FeliCa Standard card products.

INTENDED AUDIENCE

This document is written for the following readers:

- Service providers who use FeliCa card products
- Design Engineers of Reader/Writer software (firmware) for FeliCa card products
- Design Engineers of applications for the host controller for FeliCa card products, which communicates with card products via a Reader/Writer
- Middleware Design Engineers who develop SDK or similar for FeliCa card products.

Readers of this document are assumed to have enough knowledge of software development for FeliCa technology, acquired from documents concerning card products, Reader/Writer, and SDK.

PURPOSE

The purpose of this document is to describe the basic information to be considered when developing applications and middleware that use FeliCa technology, also when developing Reader/Writer software (firmware).

This document indicates where specifications differ between card products, under headings that include the word, "Cautions". These cautions, however, do not cover all the differences between all the card products.

Even if you strictly observe these cautions, flawless operation of the application is not guaranteed.

> **Note:**
>
> Eventually you must verify the correct operation of the application at system level (i.e., in an operational environment) by using the card product.

APPLICABLE PRODUCTS

The targets of description in this document are FeliCa-Standard contactless IC cards and IC chip products.

For specific product names, see the following website:
http://www.sony.net/Products/felica/business/tech-support/index.html

Some information in this document is described from the standpoint of compatibility of applications that include mobile FeliCa IC chips. Not all the functions of mobile FeliCa IC chips, however, are explained in this document. If you have any questions concerning the development of application software that is compatible with (i.e., supports) mobile FeliCa cards, please contact FeliCa Networks, Inc. (info-fn@FeliCaNetworks.co.jp).

In this document, FeliCa products (including mobile FeliCa products) are referred to collectively as card products, except when a mobile FeliCa product is clearly specified.

PROVISION OF ASSOCIATED INFORMATION

You can find the latest version of this document and related documents at the following website:
http://www.sony.net/Products/felica/business/tech-support/index.html

Before starting the development of applications or FeliCa card-related software, download the latest information for reference from the website. If you have any questions about any specific product, please contact the supplier of that product.

Contents

(Blank page)

1 Cautions on command specifications

1.1 Product-specific specifications

[Users to be especially cautious in this section]

Application Design Engineers
FeliCa Service providers

This section clearly describes the specifications that support all the card products to be the targets of this document. When you develop software not specific to the card products to be used, you are recommended to design the software while taking each item in the following list into consideration:

(1) Supported commands

FeliCa Standard card products support the following commands (excluding issuance-related commands):
If it receives an unsupported command, the card product either does not respond or returns an error.

- Polling
- Request Service
- Request Response
- Read Without Encryption
- Write Without Encryption
- Authentication1
- Authentication2
- Read
- Write
- Request System Code

(2) System Separation function

FeliCa card products support the System Separation function. The compatible number of card separation differs between individual card products. The number of card separation supported by all types of card products is 2.

(3) Number of effective blocks

The user-available size of the non-volatile memory differs between individual card products.

The specifications of the user-available size of non-volatile memory are as follows:

- RC-S860/885 150 blocks
- RC-S954 series 243 blocks
- RC-S880 396 blocks

The true number of user-available blocks differs, depending on whether the card is logically separated, the number of registered Areas and Services, and so on.

(4) Number of blocks that can be accessed simultaneously by the Read Without Encryption and the Write Without Encryption command

Number of blocks simultaneously accessible differs between individual card products.

The specifications for simultaneously-accessible blocks of all types of FeliCa card products are as follows:

Write command:	8 blocks
Write Without Encryption command:	8 blocks
Read command:	12 blocks
Read Without Encryption command:	12 blocks

(5) Communication data rate

FeliCa card products support the mutually-compatible communication data rate of 212 kbit/s.

You can check whether a card product supports a communication data rate of 424 kbit/s , by using the Polling command.

For more details, see "1.3.2 Request Code" in this document.

(6) Length of Area / Service Code

The specification supported by all types of card products is 2-Byte Area / Service Code. When transmitting a command for a 4-Byte Area or Service Code to a card product that does not support a 4-Byte Area or Service Code, the card product either does not respond or returns an error.

(7) PIN Service

PIN Service is an optional specification. When transmitting a command that tries to access the PIN Service of a card product that does not support PIN Service, the card product either does not respond or returns an error.

(As of January 2010, mobile FeliCa IC chip supports PIN Service.)

(8) Key length of Area and Service

The specification supported by all FeliCa card products is 8-Byte key.

1.2 Cautions on command packet structure

[Users to be especially cautious in this section]

Application Design Engineers
Middleware Design Engineers
Reader/writer Design Engineers

When structuring a card command packet, set only the packet data length and the value (as defined in the relevant user documentation) to the packet data area. When performing the setup described in the following cases, some card products might properly execute the command, while others might either not respond or return an error.

- The command packet has a data length longer than the defined one (caused by, for example, the addition of unnecessary data)
- The command packet has a command data length shorter than the defined one (caused when, for example, necessary data is missing)
- An undefined value is set to the command packet (undefined parameter, undefined data, and so on)

On Padding Data:

Padding Data shall be set in a manner so that the encrypted portion becomes a multiple of "8".

The length of Padding Data to be added in encrypting a command packet shall be in a range of 0-Byte to 7-Byte. If the length of Padding Data is equal to or greater than 8-Byte, it might be regarded as unnecessary data and there might be no response returned.

Do not add Padding Data to the non-encrypted command packet.

1.3 Cautions on the Polling command

[Users to be especially cautious in this section]

Application Design Engineers
Middleware Design Engineers
Reader/writer Design Engineers

1.3.1 Setting a wildcard for System Code

For the System Code of the Polling command, you can set a wildcard (ffh) on a Byte-by-Byte basis. In the course of comparison with the System Code of System existing in a card, the Byte to which the wildcard was specified is regarded as having any value. Therefore, using a wildcard you can capture several cards that have different System Codes, with only a single execution of the Polling command.. Therefore, using a wildcard you can capture several cards that have different System Codes, with only a single execution of the Polling command.

When the System Code of a logical card (System) is "0123h", for example, the card returns a response for System Code setup values "0123h", "ff23h", "01ffh", or "ffffh". Additionally, for a card that has two or more systems, comparison is performed sequentially starting from logical card 0 (System 0). Only the logical card (System) of which System Code is first detected returns a response. If you want to retrieve a specific logical card (System), use the Request System Code command.

When setting "ffffh" as the wildcard, all the cards can return a response and thereby significantly increase the probability of collision occurrence among responses returned simultaneously from more than one card. Where the application can identify the System Code of the card, avoid settng up a wildcard. Instead, you are recommended to execute the Polling command while setting a specific value to System Code.

1.3.2 Request Code

FeliCa card products support the request codes shown in Table 1-1. For FeliCa Standard, depending on which card products you use, the corresponding Request Code differs. When specifying a non-corresponding Request Code, no Request Data (2-Byte) is added to the Polling response. The design of the application shall be performed assuming that there are cases where no Request Data is added, even when specifying a Request Code.

Table 1-1: Supported request codes

Request Code	Content	Description on Request Data (2 Byte)
00h	No Request Data	No additional data
01h	Request for System Code	System Code existing in the card product, or:, for a card that has two or more systems, the first matching System Code, searching from System 0
02h	Request for communication performance	1st Byte: 00h 2nd Byte: If the card supports this code, the applicable bit becomes "1". b7(MSB): Supports automatic switching of the communication data rate[1] b6-b2: Reserved (0) b1: Supports 424 kbit/s b0: Supports 212 kbit/s
Other	Reserved (at the time of publication, no applicable product exists)	No additional data

[1] After the Polling command, data processing is performed by automatically discriminating the communication data rate of commands received from the Reader/Writer.

Note:

Data reserved at the time this document was published might not continue to be reserved. Therefore, do not check the value of reserved data with either the Reader/Writer or the application.

1.3.3 Setting up Time Slot

For the time slot values to be set for the Polling command, specify only the prescribed values (00h, 01h, 03h, 07h, or 0fh).

If values other than the prescribed ones are specified, the results can differ, depending on which card products you use.

1.4 Cautions on Read-Write commands

[Users to be especially cautious in this section]

Application Design Engineers
Middleware Design Engineers

1.4.1 Service Code List of Read Without Encryption and Write Without Encryption commands

For the Service Code List, specify only the Service Code existing in the card product. Even when a Service Code exists in the card product, do not specify a Service Code not referenced from the Block List to the Service Code List.

To detect a Service in a card product, use the Request Service command.

1.4.2 Block List Element

For each Block List, there are two methods to specify a Block; that is, the 2-Byte Block List Element specification method and the 3-Byte Block List Element specification method. Mixed use of these methods in the same Block List is possible. Middleware must be designed to support data lengths of both 2-Byte and 3-Byte.

1.4.3 Data accessible with Read Without Encryption and Write Without Encryption commands

Use the Read Without Encryption command and the Write Without Encryption command to read and write Block Data of a Service without mutual authentication between a card product and a Reader/Writer. Any Service accessible with this command is not protected by a key. Therefore, anyone can read or write such a Service. In addition, the communication path is not encrypted.

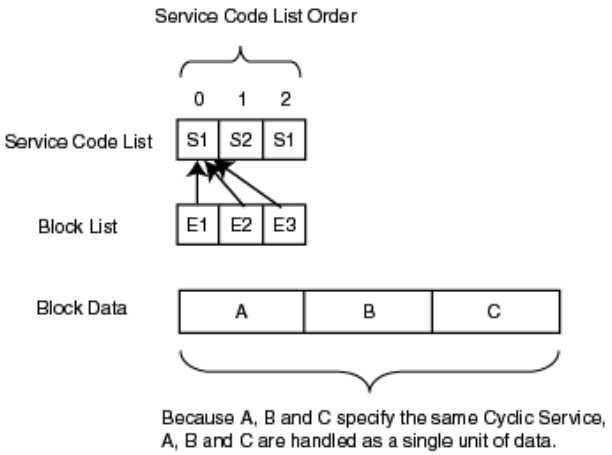
When you use this command, please take the characteristics of the application into consideration.

1.4.4 Comparison of multiple Block Data in Cyclic Service

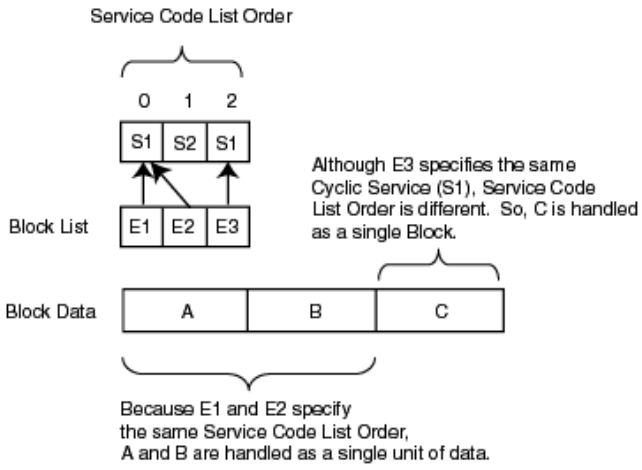
In Cyclic Service, Block Data in the Service of the write destination is compared with the data that is to be written before starting the data write process. When these two sets of data are exactly the same, Block Data is not updated, to prevent loss of data due to repeated writing of the same data.

Note:

When multiple data are written sequentially, comparison of data before data writing is performed simultaneously between multiple blocks while regarding multiple data as a single set of data. Data is regarded as a single set of data when every Block List Element that specifies data writing to that Cyclic Service are continuous and, at the same time, the Service Code List Order of each Block List Element are identical. Therefore, different data are written to Cyclic Service with access methods, as shown on the left and right of the following illustration:



In the case where the data of Cyclic Service located in a card are C, B, A (the latest one comes first), no new Block Data is written.



Even in the case where the data of Cyclic Service located in a card are C, B, A (the latest one comes first), new Block Data is written.

1.5 Timeout and Retry control

[Users to be especially cautious in this section]

Application Design Engineers
Middleware Design Engineers
Reader/Writer Design Engineers

The command-processing time in card products differs depending on the commands and different card products in use. By capturing the command-processing time and then performing the Retry process at the appropriate times, it becomes possible to perform stable RF communication.

The command-processing time of each card product can be acquired from PMm data returned in the Polling response.

You are recommended to calculate the maximum command response time based on PMm data, and then to perform the appropriate Retry processes.

1.6 Interrupt process of card commands

[Users to be especially cautious in this section]

Middleware Application Design Engineers

While a card is processing a command, it cannot receive (and so it ignores) any new command. After completing the processing of the command and transmitting the response data, the card returns to its 'listening' state.

If you want to interrupt communication with the card before transmitting a new command, either wait until a response is returned from the card, or wait until the maximum response time of the card (this can be calculated from PMm) has elapsed[1].

[EXAMPLE OF CARD COMMAND INTERRUPTION VIA A READER/WRITER]

Some Reader/Writer products might be equipped with a 'command interrupt' function. The application, however, is unable to stop processing the command in a card using such a function of the Reader/Writer.

Therefore, even if another card command is transmitted to the Reader/Writer immediately after the application used the 'command interrupt' function of the Reader/Writer, such command might not be executed in some cases (see Figure 1-1). In other cases, an unexpected response might be returned by the Reader/Writer (see Figure 1-2).

After issuing the command interrupt instruction to the Reader/Writer, it is recommended that the application waits for a period of [maximum response time of the interrupted card command + response communication time from the card to the Reader/Writer]. After the recommended period of time has elapsed, the next card command may be transmitted (see Figure 1-3).

You can calculate the maximum response time of a card based on the PMm value obtained from the the Polling response. The response communication time from a card to the appropriate Reader/Writer depends on the response packet length. The maximum response communication time is approximately 10ms.

<Caution>

The following examples of the sequence discussed in this chapter are based on specific circumstances, but the sequence can differ, depending on the available functions of the Reader/Writer in use. For example, some Reader/Writers have a 'command retry' function for cards. Even if a specific sequence is not specified in this section, such a Reader/Writer can restore the regular sequence of operations only by executing the command retry function.

You are recommended to design your application after the available functions of the appropriate Reader/Writer are checked and confirmed.

[1] You can calculate the maximum response time for the Polling command from the setup time of Time Slot, not from PMm.

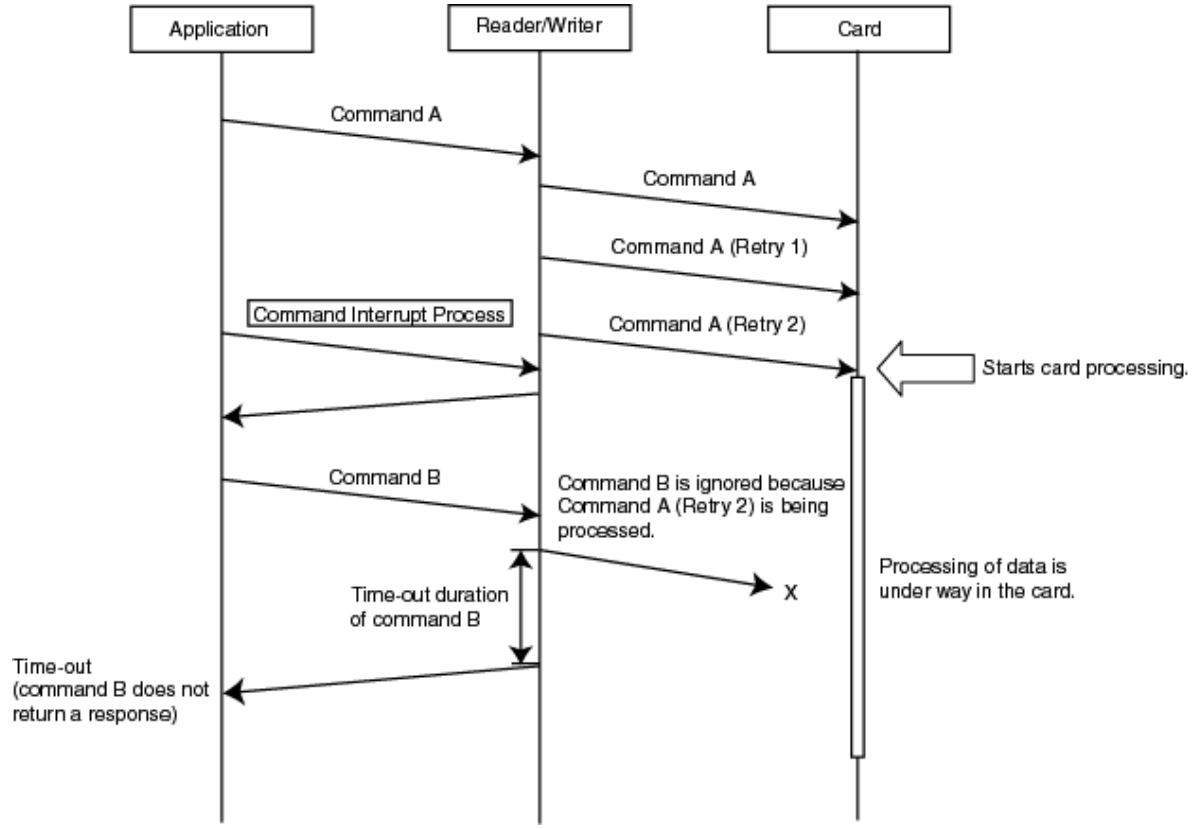


Figure 1-1 : Example sequence (where no response is returned)

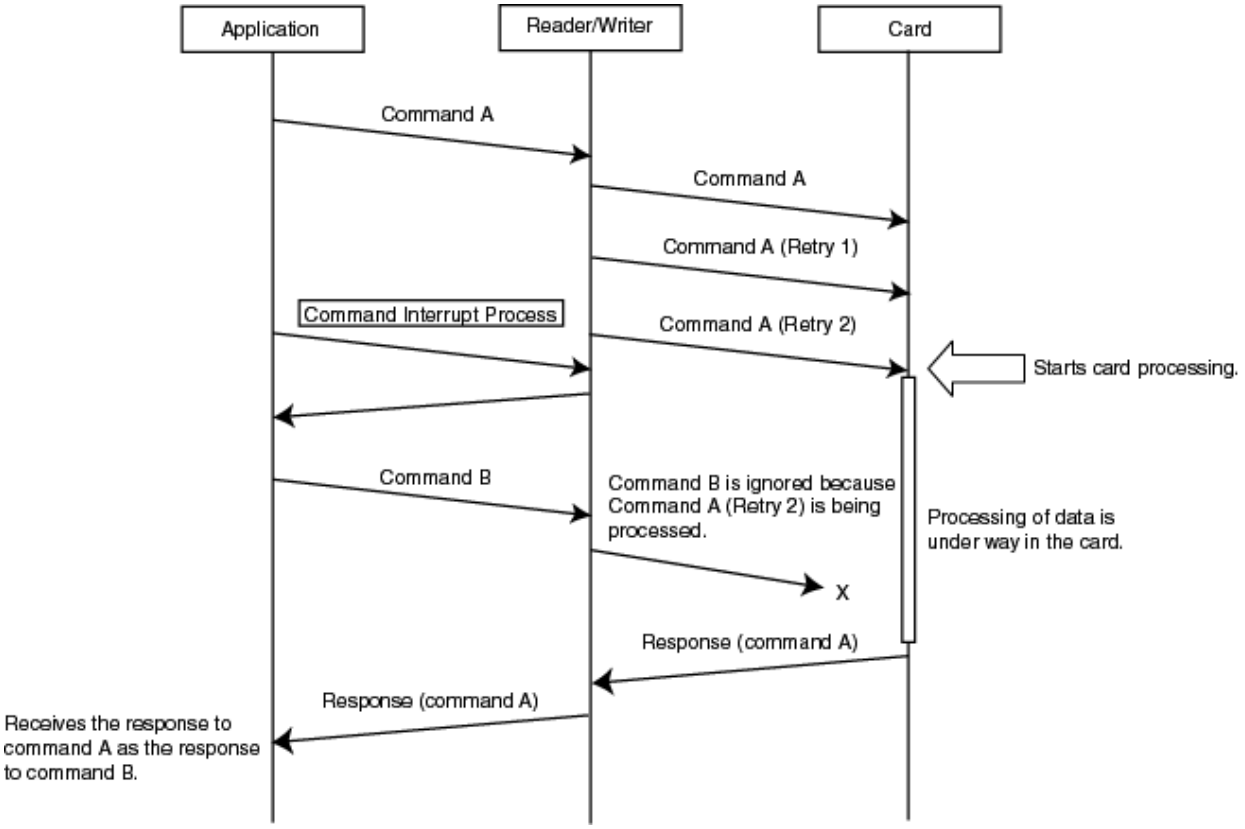


Figure 1-2: Example sequence (where an unexpected response is returned)

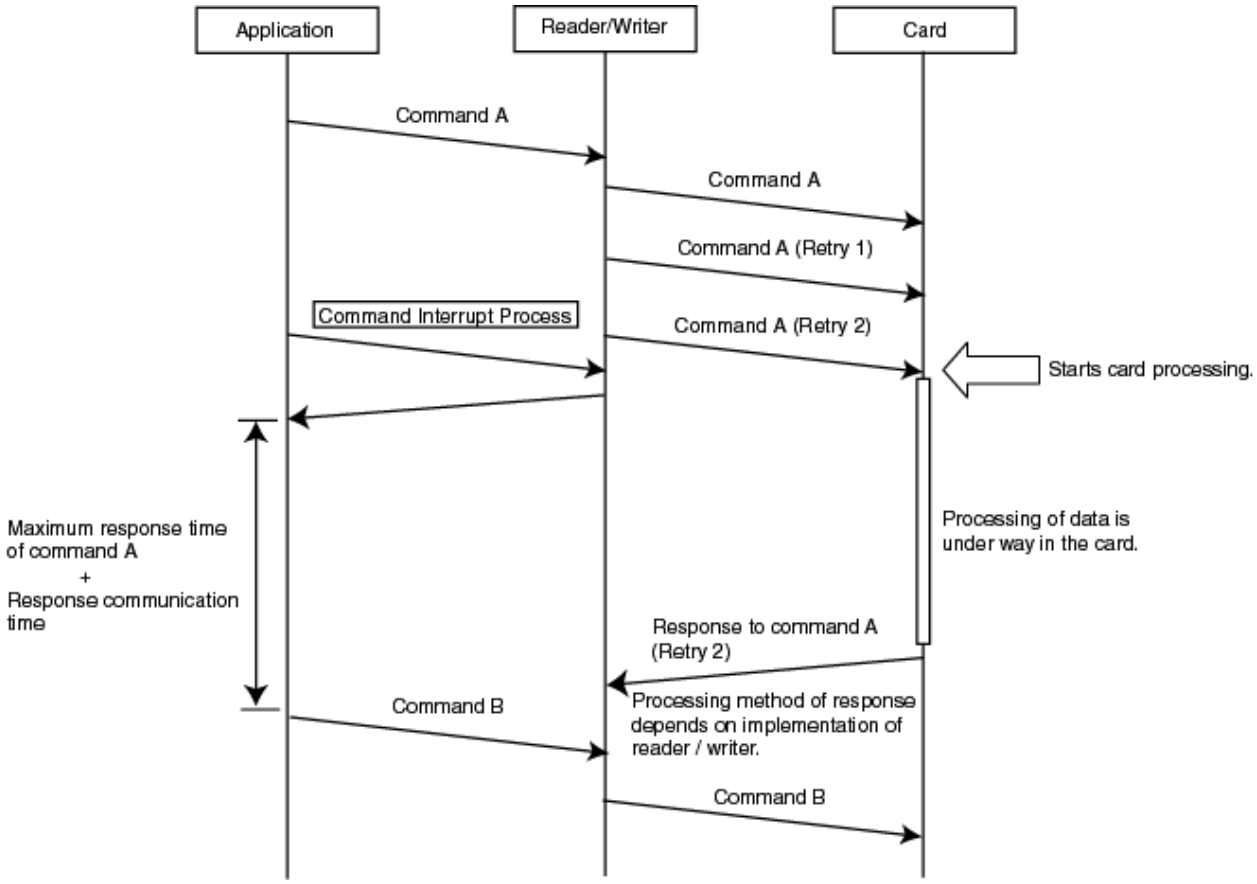


Figure 1-3: Example sequence (expected behaviour)

2 Cautions on RF interface

2.1 Cautions on magnetic field control

[Users to be especially cautious in this section]

Reader/writer Design Engineers
Application Design Engineers

When a card product enters the magnetic field emanating from the Reader/Writer, the card product initializes the operation of the IC in the card. After the initialization process completes, the card product enters its listening state.

Taking the start-up time of the magnetic field into consideration , it is recommended that the Reader/Writer continues the emanation of the magnetic field for at least 24ms. Afterwards, the Reader/Writer shall transmit a Polling command. The Reader/Writer can transmit the Polling command before 24ms has elapsed. In such cases, however, you are recommended to retry transmitting the Polling command, while taking into consideration the case where the card product fails to return a response to the first Polling command.

2.2 Guard time

[Users to be especially cautious in this section]

Reader/writer Design Engineers

After receiving a response from a card product, the Reader/Writer application shall transmit the next command after more than 501μS has elapsed.

After transmitting a command to a card product, the Reader/Writer shall enter its listening state, which is able to receive a response from the card within 198μS.

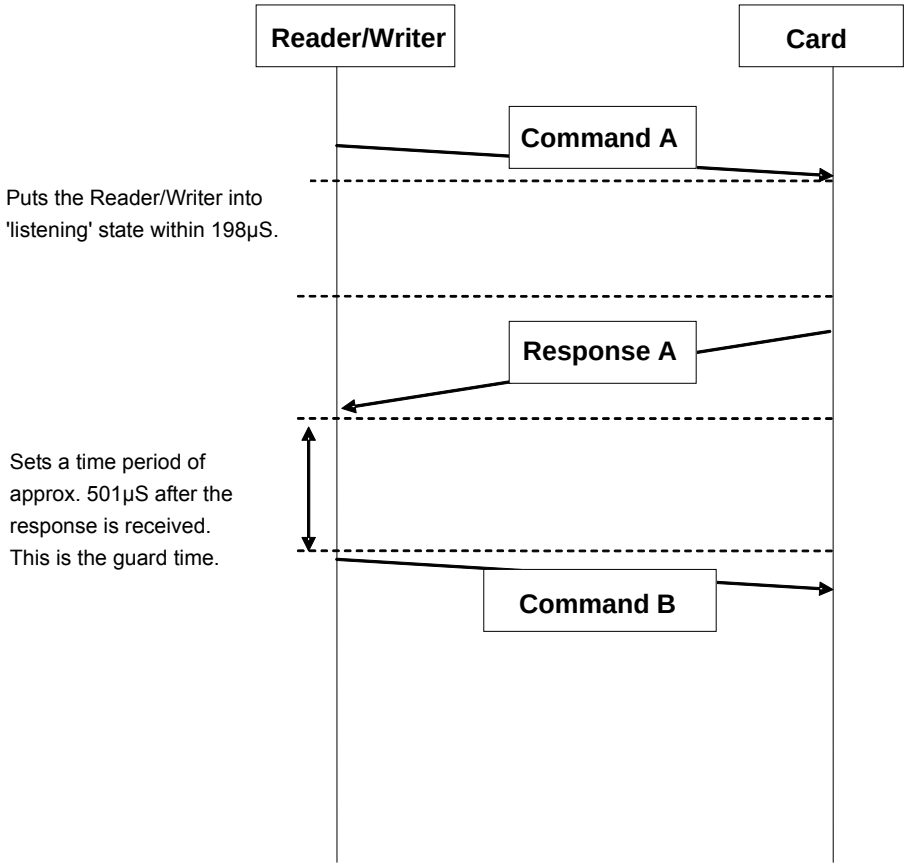


Figure 2-1： Guard time for Reader/Writer application and for card product

3 Cautions on mutual authentication

3.1 Cautions on Area and Service to be mutually authenticated

[Users to be especially cautious in this section]

Application Design Engineers

At the time of mutual authentication, the following conditions shall be observed in specifying Area Code List and Service Code List.

If any one (or more) of the conditions enumerated in the following list are not satisfied, no response might be returned, depending on which card products you use:

<Conditions for Area Code List>

- Codes in the list shall be the ones existing in the System.
- The code to be specified to the top of the list shall be Area Code.
- The code to be specified shall be Area Code or a code that indicates System (i.e., not Service Code).
- The list shall include Area Code located in the code hierarchy higher by one level than the area included in Area Code List, and also shall include Area Code located in the code hierarchy higher by one level than the Service included in Service Code List.[2]
  (As a result, the list includes Areas located in higher levels of code hierarchy up to Area 0 in chains.)

<Conditions for Service Code List>

- Codes in the list shall be the ones existing in the System.
- The list shall include Service Code of Service "with security".
- Service Code of Service "without security" shall be specified behind all the Service Codes of Service "with security".
- Where Service "with security" or Area "with security" exists under each Areas in the Area Code List, at least one such Service or Area shall be included.[2]

[2] These conditions are not mandatory for applications that target card products other than mobile FeliCa products. They are, however, mandatory when designing applications that target mobile FeliCa products.

3.2 Cautions on operation of Authentication2 command

[Users to be especially cautious in this section]

Application Design Engineers
Middleware Design Engineers
Reader/writer Design Engineers

After mutual authentication completes, do not execute the Authentication2 command to check for the existence of a card product. Instead, to detect a card product, use either the Read command or the Request Response command.

4 Cautions on Area/Service registration

4.1 Order of registration

[Users to be especially cautious in this section]

Application Design Engineers for Card Issuance

For the appropriate registration of each Area and Service, perform the registration in the following order:

(1) Perform the registration of Issue ID (IDi) using the Register Issue ID command or the Register Issue ID EX command.
(2) Perform the registration of Areas using the Register Area command.
(3) Perform the registration of Services using the Register Service command.

Performing the registration in an order other than this might return either no response or an error, depending on which card products you use.

4.2 Designation of End Service Code

[Users to be especially cautious in this section]

Application Design Engineers for Card Issuance

Set the lower 6 Bits of End Service Code of Area Package to "111111b". As an exception, however, set the lower 6 Bits to "111110b" in the case where the upper 10 Bits of End Service Code are all "1".

When specifying a value other than "111111b", it might be modified automatically to "111111b", depending on which card products you use.

4.3 Cautions on registration of Overlap Service

[Users to be especially cautious in this section]

Application Design Engineers for Card Issuance

When performing the registration of Overlap Service, set the number of User Blocks of the Service to be registered and the number of User Blocks of Service in the overlap destination to the same values.

If the number of User Blocks differ, some card products register Service by automatically modifying the numbers of User Blocks to the number of User Blocks of Service in the overlap destination. In other card products, the registration process might return either no response or an error.

4.4 Cautions on Service registration

[Users to be especially cautious in this section]

Application Design Engineers for Card Issuance

To register Service, set the number of Blocks to be registered to a value of at least "1".

If "0" is specified as the number of Blocks, it might cause either no response or an error to be returned, depending on which card products you use.

4.5 Cautions on registration of Service with large Block size

[Users to be especially cautious in this section]

Application Design Engineers for Card Issuance

When regiistering a Service containing a large number of Blocks, the registration process might not complete within the maximum response time calculated from PMm, depending on which card products you use.

If for any reason no response is returned during Service registration, and it is not known whether the registration process was successful, you can check the registration result by retrying the Registration command. If the registration was successful, a response of "Success" is returned to the retried Registration command received after completion of the registration process.

Recommended values of the retry interval for Registration commands are as follows:

Tct [ms] = (maximum response time of card calculated based on PMm) x (quotient of (Nrs / 16) + 1)

where:

- Tct: is the recommended value for retry interval
- Nrs: is the number of Blocks to be registered.

5 Cautions on using Status Flag

5.1 Use of common specifications

[Users to be especially cautious in this section]

Application Design Engineers
Reader/writer Design Engineers

During operation, if both Status Flag1 and Status Flag2 are "00h" you are recommended to determine that the operation successfully completed. In other cases, you are recommended to respond appropriately, depending on the values of Status Flag in use.

When determining the occurrence of any error, based on the values of Status Flag, you are recommended to use 80h - ffh of Status Flag2 (these are specifications unique to the card) only for debugging purposes during application development .

Meanings of 00h - 7fh of Status Flag2 are common for each card product. The methods of error determination differ internally for each card product, however, so the behavior (i.e., value) can differ even if it is the same condition as seen from the system level.

5.2 Excessive attempts to write data to non-volatile memory

[Users to be especially cautious in this section]

Application Design Engineers
FeliCa Service providers

When the value of Status Flag2 is 71h, it indicates a warning that the maximum number of data-writes to non-volatile memory has been exceeded. Although the process of writing data to non-volatile memory completes successfully, in this case, there is a possibility that the value of the written data is not maintained. Therefore, you are recommended to respond appropriately to the characteristics of the application, such as by encouraging the user to replace the card product.

Note:

Status Flag1 may have values of "00h" or "ffh".

6 Cautions on Key

6.1 Cautions on Key Version

[Users to be especially cautious in this section]

Application Design Engineers

Note:

When setting "ffffh" as the Key Version of Area / Service, it becomes impossible to check for the existence of the Area / Service using the Request Service command.

6.2 Cautions on changing Key

[Users to be especially cautious in this section]

Application Design Engineers

In a single Write command packet, avoid changing the key of the same Area or Service more than once.

When specifying such key changes, the behavior of the card can differ, depending on which card products you use.

For Key Version after the key change, set a value different from that of the current Key Version.

If any error occurs after the key change, check Key Version using the Request Service command. If Key Version is identical to the one before the key change, perform the process of key change again.

A Appendix

A.1 Status Flag

Status Flag is made up of "Status Flag 1" (1-Byte) and "Status Flag 2" (1-Byte).

Status Flag1 indicates the status (i.e., success or failure) of the process executed in the card product, and the address of the Block or Service in which an error might have occurred. Status Flag2 indicates the detailed contents of Status Flag1.

Status Flag1

Status Flag1 indicates the status (i.e., success or failure) of the process executed in the card product, and the address of the Block or Service in which an error might have occurred, as follows:

- 00h: Indicates successful completion of process execution, or warning.
- ffh: Indicates occurrence of an error independent of Block List or Service Code List, or warning. (For details, see Status Flag2.)
- xxh: Indicates that an error occurred at so-and-so number in the Block List.[3] (For details, see Status Flag2.)

[Indicates the location of error occurrence with the order in lists]

If the processing of a command fails, an error is returned in the Response packet. In this case, the location of Block, or the location of Service, or the location of the setup value string is set to Status Flag1 and returned as the response. The type of error is indicated in Status Flag2.

[Indicates the location of error occurrence with bit data]

The following list indicates the content of Status Flag1, as viewed on a bit-by-bit basis:

- Bit 0: The 1st or the 9th location in Block List or in Service Code List
- Bit1: The 2nd or 10th location in Block List or in Service Code List
- Bit2: The 3rd or 11th location in Block List or in Service Code List
- Bit3: The 4th or 12th location in Block List or in Service Code List
- Bit4: The 5th or 13th location in Block List or in Service Code List
- Bit5: The 6th or 14th location in Block List or in Service Code List
- Bit6: The 7th or 15th location in Block List or in Service Code List
- Bit7: The 8th location in Block List or in Service Code List

[3] There are two formats for indication of error occurrence depending on card products.

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

- 0: No error occurred.
- 1: An error was detected.

Block List or Service Code List is checked sequentially from the 1st location to the 12th location. The location at which an error was detected for the first time is determined as the location of error occurrence. Status Flag2 indicates the error information for such a Block. Errors located after this Block are not checked.

(Example) Where Status Flag1 = 04h:

- The 1st location in Block List or in Service Code List: no error
- The 2nd location in Block List or in Service Code List: no error
- The 3rd location in Block List or in Service Code List: an error occurred. (Detailed information is stored to Status Flag2.)
- The 4th location in Block List or in Service Code List: not inspected.
- The 5th location in Block List or in Service Code List: not inspected.
- The 6th location in Block List or in Service Code List: not inspected.
- The 7th location in Block List or in Service Code List: not inspected.
- The 8th location in Block List or in Service Code List: not inspected.

Status Flag 2

Status Flag2 indicates the detailed contents of Status Flag1. Status Flag2 is divided into two groups: Common Specifications and Specifications Unique to Card. System Design Engineers are requested to configure System using only the information of common specifications, and not to install the information based on card-specific specifications, which should be used only for System debugging purposes.

Common specifications (01h - 7fh)

00h: Processing of data completed successfully.

01h: Either (at the time of the Purse Decrement process) the result of the calculation is less than zero, or (at the time of the Purse Cash-back process) the result of the calculation is a number that exceeds 4 Bytes in length.

02h: The specified Purse data exceeded the value of the Cash-back data.

50h: An illegal PIN was specified.

51h: The number of attempts to verify the PIN exceeded the maximum limit.

52h: The number of simultaneous release attempts of PIN exceeded the maximum number.[4]

53h: PIN verification is necessary.

70h: A memory error (fatal error) occurred.

71h: The number of memory rewrite attempts exceeded the maximum limit (this is a warning, and the data write process is performed)

The maximum limit for rewriting memory differs between card products. Also, Status Flag1of some card products is "00h", but in others, Status Flag1 is "ffh".

[4] The upper limit for the simultaneous release of PIN complies with the chip specifications of the card used.

Specifications unique to card (80h - ffh)

The following list shows the status codes of major errors, for verification of application:

Notes:

- The causes of errors differ somewhat between card products, so you are recommended not to use these specifications to determine errors during proper operation.
- Use the following specifications only to debug your application:
  - a1h: Illegal number of Services.
  - a2h: Illegal command packet (number of Blocks).
  - a3h: Illegal Block List (Service Code List Order).
  - a4h: Illegal Service Type.
  - a5h: Access is inhibited.
  - a6h: Illegal Service Code List.
  - a7h: Illegal Block List (Access Mode).
  - a8h: Illegal Block Number (access to the referenced data is inhibited).
  - a9h: Data write failed.
  - aah: Key change failed.
  - abh: Illegal Parity.
  - ach: Illegal parameter.
  - adh: Service exists already.
  - aeh: Illegal System Code.
  - afh: Too much simultaneous Cyclic Write.

(Blank page)

Technical Information

FeliCa Card Technical Note for Software Development Version 1.01

July 2010 First Edition FeliCa Business Division

Sony Corporation

No. M606-E01-01

Printed in Japan