ACR122S
Serial NFC Reader

Communication Protocol V2.02
Table of Contents

1.0. Introduction ............................................................................................................. 4

2.0. Features ................................................................................................................... 5

2.1. Serial Interface ......................................................................................................... 5

2.2. Bi-color LED ............................................................................................................. 5

2.3. Buzzer ....................................................................................................................... 6

2.4. SAM Interface .......................................................................................................... 6

2.5. Built-in Antenna ....................................................................................................... 6

3.0. Communication between contactless interface and peripherals ......................... 7

4.0. Serial Interface (CCID-like Frame Format) ............................................................. 8

4.1. Protocol Flow Examples ........................................................................................ 9

5.0. SAM Interface ........................................................................................................ 11

5.1. Activating the SAM interface ................................................................................ 11

5.2. Deactivating the SAM interface ............................................................................ 12

5.3. Exchanging data through the SAM interface ....................................................... 13

6.0. Pseudo-APDUs for contactless interface and peripherals control .................... 15

6.1. Direct Transmit .................................................................................................... 15

6.2. Change Communication Speed .......................................................................... 18

6.3. Get Firmware Version ........................................................................................ 22

6.4. Bi-color LED and Buzzer Control ........................................................................ 23

6.5. Topaz512 and Jewel96 .......................................................................................... 28

6.6. Basic program flow for ISO 14443-4 Type A and B tags .................................... 35

6.7. Basic program flow for MIFARE applications .................................................... 37

6.7.1. Handling the value blocks of MIFARE 1K/4K tag? ......................................... 39

6.7.2. Accessing MIFARE Ultralight tags ................................................................. 42

6.7.3. Accessing MIFARE Ultralight C tag .............................................................. 44

6.8. Basic program flow for FeliCa applications ....................................................... 49

6.9. Basic program flow for NFC Forum Type 1 tag applications ............................. 49

Appendix A. Topaz ...................................................................................................... 51

Appendix B. Topaz512 ............................................................................................... 52

Appendix C. Jewel64 .................................................................................................... 54

Appendix D. Jewel96 .................................................................................................... 55

Appendix E. ACR122 Error Codes ............................................................................ 56

List of Figures

Figure 1 : ACR122S Communication Flowchart ...................................................... 7

Figure 2 : Tag Address “ADD” ................................................................................ 33

Figure 3 : Tag Address “ADD8” .............................................................................. 34
List of Tables

Table 1: PIN Configuration .................................................................................................................... 5
Table 2: MIFARE 1K Memory Map ...................................................................................................... 41
Table 3: MIFARE 4K Memory Map ...................................................................................................... 41
Table 4: MIFARE Ultralight Memory Map ............................................................................................ 44
Table 5: MIFARE Ultralight C Memory Map ......................................................................................... 48
1.0. Introduction

The ACR122S is a contactless smart card reader/writer used for accessing ISO 14443-4 Type A and B, MIFARE®, ISO 18092 or NFC, and FeliCa tags using the serial interface. This document will discuss the command set in implementing a smart card application using the ACR122S.
2.0. Features

- Serial RS-232 Interface: Baud Rate = 115200 bps, 8-N-1
- USB interface for power supply
- CCID-like frame format (Binary format)
- Smart Card Reader:
  - Read/Write speed of up to 424 Kbps
  - Built-in antenna for contactless tag access, with card reading distance of up to 50 mm (depending on tag type)
  - Support for ISO 14443 Part 4 Type A and B cards, MIFARE, FeliCa, and all four types of NFC (ISO/IEC 18092 tags)
  - Built-in anti-collision feature (only one tag is accessed at any time)
  - ISO 7816-compliant SAM slot
- Built-in Peripherals:
  - Two user-controllable LEDs
  - User-controllable buzzer
- Compliant with the following standards:
  - ISO18092
  - ISO 14443
  - CE
  - FCC
  - KC
  - VCCI
  - RoHS 2

2.1. Serial Interface

The ACR122S is connected to a Host through the RS-232C Serial Interface at 9600 bps, 8-N-1.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vcc</td>
<td>+5 V power supply for the reader (max. 200 mA; normal 100 mA)</td>
</tr>
<tr>
<td>2</td>
<td>TXD</td>
<td>The signal from the reader to the host</td>
</tr>
<tr>
<td>3</td>
<td>RXD</td>
<td>The signal from the host to the reader</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>Reference voltage level for power supply</td>
</tr>
</tbody>
</table>

Table 1: PIN Configuration

2.2. Bi-color LED

A user-controllable bi-color LED with red and green color is provided.

- The green LED will blink if the “Card Interface” is not connected.
- The green LED will turn on if the “Card Interface” is connected.
- The green LED will flash if the “Card Interface” is operating.
- The red LED is controlled by the application only.
2.3. Buzzer
A user-controllable buzzer with a default state of OFF is provided.

2.4. SAM Interface
One SAM socket is provided.

2.5. Built-in Antenna
A 3-turn symmetric loop antenna, center-tapped is provided.
- Estimated size is 60 mm x 48 mm
- Loop inductance is approximately 1.6 µH – 2.5 µH
- Operating distance for a different tag is approximately up to 50 mm (depends on the tag)
- Only one tag can be accessed at a time
3.0. Communication between contactless interface and peripherals

The contactless interface and peripherals are accessed through the use of pseudo-APDUs. The SAM interface is accessed through the use of standard APDUs.

![Figure 1: ACR122S Communication Flowchart](image)
4.0. Serial Interface (CCID-like Frame Format)

**Note:** Communication setting: 9600 bps, 8-N-1.

The communication protocol between the host and ACR122S is very similar to the CCID protocol.

Command Frame Format

<table>
<thead>
<tr>
<th>STX (02h)</th>
<th>Bulk-OUT Header</th>
<th>APDU Command or Parameters</th>
<th>Checksum</th>
<th>ETX (03h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Byte</td>
<td>10 Bytes</td>
<td>M Bytes (If applicable)</td>
<td>1 Byte</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

Status Frame Format

<table>
<thead>
<tr>
<th>STX (02h)</th>
<th>Status</th>
<th>Checksum</th>
<th>ETX (03h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

Response Frame Format

<table>
<thead>
<tr>
<th>STX (02h)</th>
<th>Bulk-IN Header</th>
<th>APDU Response or abData</th>
<th>Checksum</th>
<th>ETX (03h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Byte</td>
<td>10 Bytes</td>
<td>N Bytes (If applicable)</td>
<td>1 Byte</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

Checksum = XOR {Bulk-OUT Header, APDU Command or Parameters}
Checksum = XOR {Bulk-IN Header, APDU Response or abData}

In general, we would make use of three types of Bulk-OUT Header:

- **HOST_to_RDR_IccPowerOn**: To activate the SAM interface. The ATR of the SAM will be returned if available.
- **HOST_to_RDR_IccPowerOff**: To deactivate the SAM interface.
- **HOST_to_RDR_XfrBlock**: To exchange APDUs between the host and ACR122S.

The SAM interface must be activated in order to use the contactless interface and peripherals. In short, all the APDUs are exchanged through the SAM interface.

Similarly, two types of Bulk-IN Header are used:

- **RDR_to_HOST_DataBlock**: In response to the HOST_to_RDR_IccPowerOn and HOST_to_RDR_XfrBlock Frames.
- **RDR_to_HOST_SlotStatus**: In response to the HOST_to_RDR_IccPowerOff Frame.

RDR = ACR122S; HOST = Host Controller
HOST_to_RDR = Host Controller -> ACR122S
RDR_to_HOST = ACR122S -> Host Controller
4.1. Protocol Flow Examples

A. Activate a SAM.

<table>
<thead>
<tr>
<th>HOST</th>
<th>RDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. HOST sends a frame</td>
<td>02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03</td>
</tr>
<tr>
<td>2. RDR sends back a positive status frame immediately</td>
<td>02 00 00 03 (positive status frame)</td>
</tr>
<tr>
<td></td>
<td>.. after some processing delay ..</td>
</tr>
<tr>
<td>3. RDR sends back the response of the command</td>
<td>02 80 0D 00 00 00 00 01 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03</td>
</tr>
</tbody>
</table>

B. Activate a SAM (Incorrect Checksum, HOST).

<table>
<thead>
<tr>
<th>HOST</th>
<th>RDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. HOST sends a corrupted frame</td>
<td>02 62 00 00 00 00 00 01 01 00 00 [Incorrect Checksum] 03</td>
</tr>
<tr>
<td>2. RDR sends back a negative status frame immediately</td>
<td>02 FF FF 03 (negative status frame)</td>
</tr>
<tr>
<td>3. HOST sends the frame again.</td>
<td>02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03</td>
</tr>
<tr>
<td>4. RDR sends back a positive status frame immediately</td>
<td>02 00 00 03 (positive status frame)</td>
</tr>
<tr>
<td></td>
<td>.. after some processing delay ..</td>
</tr>
<tr>
<td>5. RDR sends back the response of the command</td>
<td>02 80 0D 00 00 00 00 01 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03</td>
</tr>
</tbody>
</table>

C. Activate a SAM (Incorrect Checksum, RDR).

<table>
<thead>
<tr>
<th>HOST</th>
<th>RDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. HOST sends a frame</td>
<td>02 62 00 00 00 00 00 01 01 00 00 [Checksum] 03</td>
</tr>
<tr>
<td>2. RDR sends back a positive status frame immediately</td>
<td>02 00 00 03 (positive status frame)</td>
</tr>
</tbody>
</table>
3. RDR sends back the response (corrupted) of the command

\[
\begin{array}{l}
\text{HOST} \quad \Rightarrow \\
\text{RDR} \\
02\ 80\ 0D\ 00\ 00\ 00\ 00\ 01\ 00\ 00\ 00\ 3B\ 2A \\
00\ 80\ 65\ 24\ B0\ 00\ 02\ 00\ 82\ 90\ 00 \\
[Incorrect Checksum]\ 03
\end{array}
\]

4. HOST sends a NAK frame to get the response again.

\[
\begin{array}{l}
\text{HOST} \quad \Rightarrow \\
\text{RDR} \\
02\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 03 \\
(NAK)
\end{array}
\]

5. RDR sends back the response of the command

\[
\begin{array}{l}
\text{HOST} \quad \Rightarrow \\
\text{RDR} \\
02\ 80\ 0D\ 00\ 00\ 00\ 00\ 01\ 00\ 00\ 00\ 3B\ 2A \\
00\ 80\ 65\ 24\ B0\ 00\ 02\ 00\ 82\ 90\ 00 \\
[Checksum]\ 03
\end{array}
\]

**Note:** If the frame sent by the HOST is correctly received by the RDR, a positive status frame = \{02 00 00 03\} will be sent to the HOST immediately to inform the HOST the frame is correctly received. The HOST has to wait for the response of the command. The RDR will not receive any more frames while the command is being processed.

In case of errors, a negative status frame will be sent to the HOST to indicate the frame is either corrupted or incorrectly formatted.

**Checksum Error Frame = \{02 FF FF 03\}**

**Length Error Frame = \{02 FE FE 03\}. The length dDwLength is greater than 0105h bytes.**

**ETX Error Frame = \{02 FD FD 03\}. The last byte is not equal to ETX “03h”.**

The NAK Frame is only used by the HOST to get the last response.

\{02 00 00 00 00 00 00 00 00 00 00 00 03\} // 11 zeros
5.0. SAM Interface

5.1. Activating the SAM interface

Command Frame Format

<table>
<thead>
<tr>
<th>STX (02h)</th>
<th>Bulk-OUT Header</th>
<th>Parameters</th>
<th>Checksum</th>
<th>ETX (03h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Byte</td>
<td>10 Bytes</td>
<td>0 Byte</td>
<td>1 Byte</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

HOST_to_RDR_IccPowerOn Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>62h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>00000000h</td>
<td>Message-specific data length.</td>
</tr>
<tr>
<td></td>
<td>&lt;LSB .. MSB&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command. Default=00h</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>bPowerSelect</td>
<td>1</td>
<td>00h</td>
<td>Voltage that is applied to the ICC:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>01h</td>
<td>01h – Automatic Voltage Selection</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>02h</td>
<td>02h – 3 V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>03h</td>
<td>03h – 1.8 V</td>
</tr>
<tr>
<td>8</td>
<td>abRFU</td>
<td>2</td>
<td></td>
<td>Reserved for Future Use.</td>
</tr>
</tbody>
</table>

Response Frame Format

<table>
<thead>
<tr>
<th>STX (02h)</th>
<th>Bulk-IN Header</th>
<th>abData</th>
<th>Checksum</th>
<th>ETX (03h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Byte</td>
<td>10 Bytes</td>
<td>N Bytes (ATR)</td>
<td>1 Byte</td>
<td>1 Byte</td>
</tr>
</tbody>
</table>

RDR_to_HOST_DataBlock Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>80h</td>
<td>Indicates that a data block is being sent from the ACR122S.</td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>N</td>
<td>Size of abData field (N Bytes).</td>
</tr>
<tr>
<td></td>
<td>&lt;LSB .. MSB&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>Same as Bulk-OUT</td>
<td>Identifies the slot number for this command.</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>Same as Bulk-OUT</td>
<td>Sequence number for corresponding command.</td>
</tr>
<tr>
<td>7</td>
<td>bStatus</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>bError</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Example:** To activate the slot 0 (default), sequence number = 1, 5 V card.

HOST -> 02 62 00 00 00 00 01 01 00 00 [Checksum] 03
RDR -> 02 00 00 03
RDR -> 02 80 00 00 00 01 00 00 00 3B 2A 00 80 65 24 B0 00 02 00 82 90 00 [Checksum] 03
The ATR = 3B 2A 00 80 65 24 B0 00 02 00 82; SW1 SW2 = 90 00

### 5.2. Deactivating the SAM interface

**Command Frame Format**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>63h</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>dDwLength</td>
<td>4</td>
<td>00000000h</td>
<td>Message-specific data length.</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command. Default=00h</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>abRFU</td>
<td>3</td>
<td></td>
<td>Reserved for Future Use.</td>
</tr>
</tbody>
</table>

**Response Frame Format**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>81h</td>
<td>Indicates that a data block is being sent from the ACR122S.</td>
</tr>
<tr>
<td>1</td>
<td>dwLength</td>
<td>4</td>
<td>0</td>
<td>Size of abData field (0 Byte).</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>Same as Bulk-OUT</td>
<td>Identifies the slot number for this command.</td>
</tr>
</tbody>
</table>
### Example:
To deactivate the slot 0 (default), sequence number = 2.

**HOST** -> 02 63 00 00 00 00 00 02 00 00 00 [Checksum] 03

**RDR** -> 02 00 00 03

**RDR** -> 02 81 00 00 00 00 00 02 00 00 00 [Checksum] 03

### 5.3. Exchanging data through the SAM interface

#### Command Frame Format

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>6Fh</td>
<td>Indicates that a data block is being sent from the ACR122S.</td>
</tr>
<tr>
<td>1</td>
<td>dDwLength</td>
<td>4</td>
<td>M</td>
<td>Message-specific data length.</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td>00-FFh</td>
<td>Identifies the slot number for this command.</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td>00-FFh</td>
<td>Sequence number for command.</td>
</tr>
<tr>
<td>7</td>
<td>bBWI</td>
<td>1</td>
<td>00-FFh</td>
<td>Used to extend the Block Waiting Timeout.</td>
</tr>
<tr>
<td>8</td>
<td>wLevelParameter</td>
<td>2</td>
<td>0000h</td>
<td></td>
</tr>
</tbody>
</table>

**Response Frame Format**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Field</th>
<th>Size</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bMessageType</td>
<td>1</td>
<td>80h</td>
<td>Indicates that a data block is being sent from the ACR122S.</td>
</tr>
<tr>
<td>Offset</td>
<td>Field</td>
<td>Size</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>------</td>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>dwLength &lt;LSB .. MSB&gt;</td>
<td>4</td>
<td>N</td>
<td>Size of abData field (N Bytes).</td>
</tr>
<tr>
<td>5</td>
<td>bSlot</td>
<td>1</td>
<td></td>
<td>Same as Bulk-OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Identifies the slot number for this command.</td>
</tr>
<tr>
<td>6</td>
<td>bSeq</td>
<td>1</td>
<td></td>
<td>Same as Bulk-OUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sequence number for corresponding command.</td>
</tr>
<tr>
<td>7</td>
<td>bStatus</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>bError</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>bChainParameter</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:** To send an APDU “80 84 00 00 08” to the slot 0 (default), sequence number = 3.

HOST    -> 02 6F 05 00 00 00 00 03 00 00 00 80 84 00 00 08 [Checksum] 03
RDR     -> 02 00 00 03
RDR     -> 02 80 0A 00 00 00 00 03 00 00 00 E3 51 B0 FC 88 AA 2D 18 90 00 [Checksum] 03
Response = E3 51 B0 FC 88 AA 2D 18; SW1 SW2 = 90 00
6.0. Pseudo-APDUs for contactless interface and peripherals control

ACR122S comes with two primitive commands for this purpose. <Class FFh>

6.1. Direct Transmit

This command is used to send a pseudo-APDU (Tag Commands), and returns the length of the Response Data.

Direct Transmit Command Format (Length of the Tag Command + 5 Bytes)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Data In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Transmit</td>
<td>FFh</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>Number of bytes to send</td>
<td>Tag Command</td>
</tr>
</tbody>
</table>

Where:

- **Lc**: Number of bytes to send (1 Byte)
  - Maximum 255 bytes
- **Data In**: Tag command.
  - The data to be sent to the tag.

Direct Transmit Response Format (Tag Response + Data + 2 Bytes)

<table>
<thead>
<tr>
<th>Item</th>
<th>Command</th>
<th>Data</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D4 40</td>
<td>Tg</td>
<td>[DataOut][] Tag Exchange Data</td>
</tr>
<tr>
<td>2</td>
<td>D4 4A</td>
<td>MaxTg</td>
<td>[InitiatorData][] Tag Polling</td>
</tr>
</tbody>
</table>

Where:

- **Tg**: A byte containing the logical number of the relevant target. This byte also contains the More Information (MI) bit (bit 6). When the MI bit is set to 1, this indicates that the host controller wants to send more data which is all the data contained in the DataOUT[] array. This bit is only valid for a TPE target.
- **DataOut**: An array of raw data (from 0 up to 262 bytes) to be sent to the target by the contactless chip.
- **MaxTg**: Maximum number of targets to be initialized by the contactless chip. The chip is capable of handling 2 targets maximum at once, so this field should not exceed 02h.
- **Brty**: Baud rate and the modulation type to be used during the initialization.
  - 00h: 106 kbps type A (ISO/IEC14443 Type A),
  - 01h: 212 kbps (FeliCa polling),
  - 02h: 424 kbps (FeliCa polling),
  - 03h: 106 kbps type B (ISO/IEC 14443-3B),
  - 04h: 106 kbps Innovision Jewel tag.
- **InitiatorData[]**: An array of data to be used during the initialization of the target(s). Depending on the Baud Rate specified, the content of this field is different.
106 kbps type A

The field is optional and is present only when the host controller wants to initialize a target with a known UID.

In that case, InitiatorData[] contains the UID of the card (or part of it). The UID must include the cascade tag CT if it is cascaded level 2 or 3.

Cascade Level 1

| UID1 | UID2 | UID3 | UID4 |

Cascade Level 2

| UID1 | UID2 | UID3 | UID4 | UID5 | UID6 | UID7 |

Cascade Level 3

| UID1 | UID2 | UID3 | UID4 | UID5 | UID6 | UID7 | UID8 | UID9 | UID10 |

106 kbps type B

In this case, InitiatorData[] is formatted as following:

| AFI (1byte) | [Polling Method] |

AFI The AFI (Application Family Identifier) parameter represents the type of application targeted by the device IC and is used to preselect the PICCs before the ATQB.

This field is mandatory.

Polling Method This field is optional. It indicates the approach to be used in the ISO/IEC 14443-3B initialization:

- If bit 0 = 1: Probabilistic approach (option 1) in the ISO/IEC 14443-3B initialization,
- If bit 0 = 0: Timeslot approach (option 2) in the ISO/IEC 14443-3B initialization,
- If this field is absent, the timeslot approach will be used.

212/424 kbps In that case, this field is mandatory and contains the complete payload information that should be used in the polling request command (5bytes, length bytes is excluded)

106 kbps InnoVision Jewel tag. This field is not used.

Data Out Tag Response returned by the reader.

Direct Transmit Response Format

<table>
<thead>
<tr>
<th>Response</th>
<th>Data Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>DataIn[]</td>
</tr>
<tr>
<td>D5 41</td>
<td>Status</td>
</tr>
<tr>
<td>D5 4B</td>
<td>NbTg</td>
</tr>
</tbody>
</table>
Where:

**Status**
A byte indicating if the process has been terminated successfully or not. When in either DEP or ISO/IEC 14443-4 PCD mode, this byte also indicates if NAD (Node Address) is used and if the transfer of data is not completed with bit More Information.

**DataIn**
An array of raw data (from 0 up to 262 bytes) received by the contactless chip.

**NbTg**
The number of initialized Targets (minimum 0, maximum 2 targets).

**TargetData[i]**
The “i” in TargetData[i] refers to “1” or “2”. This contains the information about the detected targets and depends on the baud rate selected. The following information is given for one target, it is repeated for each target initialized (NbTg times).

### 106 kbps Type A

<table>
<thead>
<tr>
<th>Tg</th>
<th>SENS_RES10 (2 bytes)</th>
<th>SEL_RES (1 byte)</th>
<th>NFCIDLength (1 byte)</th>
<th>NFCID1[] (NFCIDLength bytes)</th>
<th>[ATS[]] (ATSLength bytes11)</th>
</tr>
</thead>
</table>

### 106 kbps Type B

<table>
<thead>
<tr>
<th>Tg</th>
<th>ATQB Response (12 bytes)</th>
<th>ATTRIB_RES Length (1 byte)</th>
<th>ATTRIB_RES[] (ATTRIB_RES Length)</th>
</tr>
</thead>
</table>

### 212/424 kbps

<table>
<thead>
<tr>
<th>Tg</th>
<th>POL_RES length 01h (response code)</th>
<th>NFCID2t</th>
<th>Pad</th>
<th>SYST_CODE (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>1 byte</td>
<td>8 bytes</td>
<td>8 bytes</td>
</tr>
<tr>
<td>POL_RES (18 or 20 bytes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 106 kbps Innovision Jewel tag

<table>
<thead>
<tr>
<th>Tg</th>
<th>SENS_RES (2 bytes)</th>
<th>JEWELID[] (4 bytes)</th>
</tr>
</thead>
</table>

### Data Out
SW1 SW2.
Status Code returned by the reader.

<table>
<thead>
<tr>
<th>Results</th>
<th>SW1</th>
<th>SW2</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>90</td>
<td>00h</td>
<td>The operation is completed successfully.</td>
</tr>
<tr>
<td>Error</td>
<td>63</td>
<td>00h</td>
<td>The operation is failed.</td>
</tr>
<tr>
<td>Time Out Error</td>
<td>63</td>
<td>01h</td>
<td>The TAG does not response.</td>
</tr>
</tbody>
</table>
### 6.2. Change Communication Speed

This command is used to change the baud rate.

#### Baud Rate Control Command Format (9 Bytes)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate Control</td>
<td>FFh</td>
<td>00h</td>
<td>44h</td>
<td>New Baud Rate</td>
<td>00h</td>
</tr>
</tbody>
</table>

Where:
- **P2** New Baud Rate
  - 00h = Set the new baud rate to 9600 bps.
  - 01h = Set the new baud rate to 115200 bps.

Data Out: SW1 SW2.

#### Status Code

<table>
<thead>
<tr>
<th>Results</th>
<th>SW1</th>
<th>SW2</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>90</td>
<td>Current Baud Rate</td>
<td>The operation is completed successfully.</td>
</tr>
<tr>
<td>Error</td>
<td>63</td>
<td>00h</td>
<td>The operation is failed.</td>
</tr>
</tbody>
</table>

Where:
- **SW2** Current Baud Rate
  - 00h = The current baud rate is 9600 bps.
  - 01h = The current baud rate is 115200 bps.

*Note: After the communication speed is changed successfully, the program has to adjust its communication speed to continue the rest of the data exchanges.*

The initial communication speed is determined by the existence of R12 (0 ohm).
- With R12 = 115200 bps
- Without R12 = 9600 bps (default)

**Example 1:** To initialize a FeliCa Tag (Tag Polling).

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be “FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00”

In which,
- Direct Transmit APDU = “FF 00 00 00”
- Length of the Tag Command = “09”
- Tag Command (InListPassiveTarget 212Kbps) = “D4 4A 01 01”
Tag Command (System Code Request) = "00 FF 01 00"

To send an APDU to the slot 0 (default), sequence number = 1.
HOST → 02 6F 0E 00 00 00 00 01 00 00 00
FF 00 00 00 09 D4 4A 01 01 00 FF FF 01 00
[Checksum] 03
RDR → 02 00 00 03
RDR → 02 81 1A 00 00 00 00 01 00 00 00
D5 4B 01 01 14 01 01 05 01 86 04 02 02 03 00
4B 02 4F 49 8A 8A 80 08 90 00
[Checksum] 03

The APDU Response is
"D5 4B 01 01 14 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08 90 00"

In which,
Response returned by the contactless chip = "D5 4B 01 01 14 01 01 05 01 86 04 02 02 03 00 4B 02 4F 49 8A 8A 80 08"
NFCID2t of the FeliCa Tag = "01 01 05 01 86 04 02 02"
Status Code returned by the reader = "90 00"

Example 2: To write 16 bytes data to the FeliCa Tag (Tag Write).
Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be "FF 00 00 00 23 D4 40 01 20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA"

In which,
Direct Transmit APDU = “FF 00 00 00”
Length of the Tag Command = “23”
Tag Command (InDataExchange) = “D4 40 01”
Tag Command (Write Data) = “20 08 01 01 05 01 86 04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA”.

To send an APDU to the slot 0 (default), sequence number = 2.
HOST → 02 6F 26 00 00 00 00 02 00 00 00
FF 00 00 00 21 D4 40 01 20 08 01 01 05 01 86
04 02 02 01 09 01 01 80 00 00 AA 55 AA 55 AA 55
AA 55 AA 55 AA 55 AA
[Checksum] 03
RDR -> 02 00 00 03  
RDR -> 02 81 11 00 00 00 00 02 00 00 00  
D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00  
[Checksum] 03  
The APDU Response would be “D5 41 00 0C 09 01 01 05 01 86 04 02 02 00 00 90 00”  

In which,  
Response returned by the contactless chip = “D5 41”  
Response returned by the FeliCa Tag = “00 0C 09 01 01 05 01 86 04 02 02 00 00”  
Status Code returned by the reader = “90 00”  

**Example 3:** To read 16 bytes data from the FeliCa Tag (Tag Write).  
Step 1. Issue a “Direct Transmit” APDU.  

The APDU Command should be “FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00”  

In which,  
Direct Transmit APDU = “FF 00 00 00”  
Length of the Tag Command = “13”  
Tag Command (InDataExchange) = “D4 40 01”  
Tag Command (Read Data) = “10 06 01 01 05 01 86 04 02 02 01 09 01 01 80 00”  

To send an APDU to the slot 0 (default), sequence number = 3.  
HOST -> 02 6F 18 00 00 00 00 03 00 00 00  
FF 00 00 00 13 D4 40 01 10 06 01 01 05 01 86 04  
02 02 01 09 01 01 80 00  
[Checksum] 03  
RDR -> 02 00 00 03  
RDR -> 02 81 22 00 00 00 00 03 00 00 00  
D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00  
AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00  
[Checksum] 03  
The APDU Response would be  
“D5 41 00 1D 07 01 01 05 01 86 04 02 02 00 00 01 00 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 55 AA 90 00”
Example 4: To initialize an ISO 14443-4 Type B Tag (Tag Polling).

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be “FF 00 00 00 05 D4 4A 01 03 00”

In which,

- Direct Transmit APDU = “FF 00 00 00”
- Length of the Tag Command = “05”
- Tag Command (InListPassiveTarget Type B 106Kbps) = “D4 4A 01 03 00”

To send an APDU to the slot 0 (default), sequence number = 4.
HOST -> 02 6F 0A 00 00 00 04 00 00 00
FF 00 00 00 05 D4 4A 01 03 00
[Checksum] 03
RDR -> 02 00 00 03
RDR -> 02 81 14 00 00 00 04 00 00 00
D5 41 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21
90 00 [Checksum] 03

The APDU Response is
“D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00”

In which,
- Response returned by the contactless chip = “D5 4B 01 01”
- ATQB of the Type B Tag = “50 00 01 32 F4 00 00 00 00 33 81 81”
- CRC-B = “01 21”
- Status Code returned by the reader = “90 00”

Example 5: To send an APDU to an ISO 14443-4 Type B Tag (Data Exchange).

Step 1. Issue a “Direct Transmit” APDU.

The USER APDU Command should be “00 84 00 00 08”
The Composed APDU Command should be “FF 00 00 00 08 D4 40 01 00 84 00 00 08”
In which,
Direct Transmit APDU = “FF 00 00 00”
Length of the Tag Command = “08”
Tag Command (InDataExchange) = “D4 40 01”
Tag Command (Get Challenge) = “00 84 00 00 08”

To send an APDU to the slot 0 (default), sequence number = 5.
HOST -> 02 6F 0D 00 00 00 05 00 00 00
FF 00 00 00 08 D4 40 01 00 84 00 00 08
[Checksum] 03
RDR -> 02 00 00 03
RDR -> 02 81 0F 00 00 00 00 05 00 00 00
D5 41 00 01 02 03 04 05 06 07 08 90 00 90 00
[Checksum] 03

The APDU Response is “D5 41 00 0B 01 02 03 04 05 06 07 08 90 00”

In which,
Response returned by the contactless chip = “D5 41 00”
Response from the Type B Tag = “01 02 03 04 05 06 07 08 90 00”
Status Code returned by the reader = “90 00”

6.3. Get Firmware Version

This command is used to derive the firmware version of the reader.

Get Firmware Version Command Format (5 Bytes)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Le</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Response</td>
<td>FFh</td>
<td>00h</td>
<td>48h</td>
<td>00h</td>
<td>00h</td>
</tr>
</tbody>
</table>

Where:
Le Number of bytes to retrieve (1 Byte).
Maximum 255 bytes.

Get Firmware Version Response Format (10 Bytes)

<table>
<thead>
<tr>
<th>Response</th>
<th>Data Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>Firmware Version</td>
</tr>
</tbody>
</table>

Example:
Response = 41 43 52 31 32 32 53 31 30 30 (Hex) = ACR122S100 (ASCII)
6.4. Bi-color LED and Buzzer Control

This command is used to control the states of the bi-color LED and buzzer.

Bi-color LED and Buzzer Control Command Format (9 Bytes)

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Data In (4 Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-color LED and Buzzer Control</td>
<td>FFh</td>
<td>00h</td>
<td>40h</td>
<td>LED State Control</td>
<td>04h</td>
<td>Blinking Duration Control</td>
</tr>
</tbody>
</table>

P2  LED State Control.

Bi-color LED and Buzzer Control Format (1 Byte)

<table>
<thead>
<tr>
<th>CMD</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>Final Red LED State</td>
<td>1 = On; 0 = Off</td>
</tr>
<tr>
<td>Bit 1</td>
<td>Final Green LED State</td>
<td>1 = On; 0 = Off</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Red LED State Mask</td>
<td>1 = Update the State; 0 = No change</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Green LED State Mask</td>
<td>1 = Update the State; 0 = No change</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Initial Red LED Blinking State</td>
<td>1 = On; 0 = Off</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Initial Green LED Blinking State</td>
<td>1 = On; 0 = Off</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Red LED Blinking Mask</td>
<td>1 = Blink; 0 = Not Blink</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Green LED Blinking Mask</td>
<td>1 = Blink; 0 = Not Blink</td>
</tr>
</tbody>
</table>

Data In  Blinking Duration Control.

Bi-Color LED Blinking Duration Control Format (4 Bytes)

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 Duration</td>
<td>T2 Duration</td>
<td>Number of repetition</td>
<td>Link to Buzzer</td>
</tr>
<tr>
<td>Initial Blinking State (Unit = 100 ms)</td>
<td>Toggle Blinking State (Unit = 100 ms)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where:

 byte 3  Link to Buzzer. Control the buzzer state during the LED Blinking

00h = The buzzer will not turn on.
01h = The buzzer will turn on during the T1 Duration.
02h = The buzzer will turn on during the T2 Duration.
03h = The buzzer will turn on during the T1 and T2 Duration.
Data Out  SW1 SW2. Status Code returned by the reader.

Status Code

<table>
<thead>
<tr>
<th>Results</th>
<th>SW1</th>
<th>SW2</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>90</td>
<td>Current LED State</td>
<td>The operation is completed successfully.</td>
</tr>
<tr>
<td>Error</td>
<td>63</td>
<td>00h</td>
<td>The operation is failed.</td>
</tr>
</tbody>
</table>

Current LED State (1 Byte)

<table>
<thead>
<tr>
<th>Status</th>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>Current Red LED</td>
<td>1 = On; 0 = Off</td>
</tr>
<tr>
<td>Bit 1</td>
<td>Current Green LED</td>
<td>1 = On; 0 = Off</td>
</tr>
<tr>
<td>Bits 2 – 7</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Notes:

1. The LED State operation will be performed after the LED Blinking operation is completed.
2. The LED will not change if the corresponding LED Mask is not enabled.
3. The LED will not blink if the corresponding LED Blinking Mask is not enabled. Also, the number of repetition must be greater than zero.
4. T1 and T2 duration parameters are used for controlling the duty cycle of LED blinking and Buzzer Turn-On duration. For example, if T1=1 and T2=1, the duty cycle = 50%. Duty Cycle = \( \frac{T1}{T1 + T2} \).
5. To control the buzzer only, set the P2 “LED State Control” to zero.
6. The make the buzzer operate, the “number of repetition” must be greater than zero.
7. To control the LED only, set the parameter “Link to Buzzer” to zero.

Example 1: To read the existing LED State.

// Assume both Red and Green LEDs are OFF initially //
// Not linked to the buzzer //

APDU = “FF 00 40 00 04 00 00 00 00”
Response = “90 00”. RED and Green LEDs are OFF.

Example 2: To turn on RED and Green Color LEDs.

// Assume both Red and Green LEDs are OFF initially //
// Not linked to the buzzer //

APDU = “FF 00 40 0F 04 00 00 00 00”
Response = “90 03”. RED and Green LEDs are ON,
To turn off both RED and Green LEDs, APDU = “FF 00 40 0C 04 00 00 00 00”
Example 3: To turn off the Red LED only, and leave the Green LED unchanged.

// Assume both Red and Green LEDs are ON initially //
// Not linked to the buzzer //

APDU = “FF 00 40 04 00 00 00 00"
Response = “90 02”. Green LED is not changed (ON); Red LED is OFF.

Example 4: To turn on the Red LED for 2 seconds. After that, resume to the initial state.

// Assume the Red LED is initially OFF, while the Green LED is initially ON. //
// The Red LED and buzzer will turn on during the T1 duration, while the Green LED will turn off during the T1 duration. //

1 Hz = 1000 ms
Time Interval = 500 ms ON + 500 ms OFF
T1 Duration = 2000 ms = 14h
T2 Duration = 0 ms = 00h
Number of repetition = 01h
Link to Buzzer = 01h
Example 5: To make the Red LED of 1 Hz blink for 3 times. After that, resume to initial state.

// Assume the Red LED is initially OFF, while the Green LED is initially ON. //
// The Initial Red LED Blinking State is ON. Only the Red LED will be blinking.
// The buzzer will turn on during the T1 duration, while the Green LED will turn off during both the T1 and T2 duration.
// After the blinking, the Green LED will turn ON. The Red LED will resume to the initial state after the blinking //

![Diagram showing LED and buzzer states]

1 Hz = 1000 ms Time Interval = 500 ms ON + 500 ms OFF
T1 Duration = 500 ms = 05h
T2 Duration = 500 ms = 05h
Number of repetition = 03h
Link to Buzzer = 01h

APDU = “FF 00 40 50 04 05 05 03 01”
Response = “90 02”

Example 6: To make the Red and Green LEDs of 1 Hz blink for 3 times.

// Assume both the Red and Green LEDs are initially OFF. //
// Both Initial Red and Green Blinking States are ON //
// The buzzer will turn on during both the T1 and T2 duration//
1 Hz = 1000 ms Time Interval = 500 ms ON + 500 ms OFF
T1 Duration = 500 ms = 05h
T2 Duration = 500 ms = 05h
Number of repetition = 03h
Link to Buzzer = 03h

APDU = “FF 00 40 F0 04 05 05 03 03”
Response = “90 00”

Example 7: To make the Red and Green LEDs in turn of 1 Hz blink for 3 times.
// Assume both Red and Green LEDs are initially OFF. //
// The Initial Red Blinking State is ON; The Initial Green Blinking States is OFF //
// The buzzer will turn on during the T1 duration//
1 Hz = 1000 ms
Time Interval = 500 ms ON + 500 ms OFF
T1 Duration = 500 ms = 05h
T2 Duration = 500 ms = 05h
Number of repetition = 03h
Link to Buzzer = 01h

APDU = "FF 00 40 D0 04 05 05 03 01"
Response = "90 00"

6.5. Topaz512 and Jewel96

*Note: This section only applies to ACR122S with firmware version 1.03.*

This command is used to Write-with-erase (8 Bytes), Write-no-erase (8 Bytes), Read (8 Bytes) and Read Segment.

**Topaz 512 and Jewel 96 Command Format**

<table>
<thead>
<tr>
<th>Command</th>
<th>Class</th>
<th>INS</th>
<th>P1</th>
<th>P2</th>
<th>Lc</th>
<th>Data In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Transmit</td>
<td>FFh</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>Number of bytes to send</td>
<td>TAG Command</td>
</tr>
</tbody>
</table>

Where:

**Lc**            Number of bytes to send (1 Byte).

Maximum 255 bytes.

**Data In**       Tag Command.

The data to be sent to the tag.
Direct Transmit Response Format (Response Length + 2 Bytes)

<table>
<thead>
<tr>
<th>Response</th>
<th>Data Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>Tag Response</td>
</tr>
</tbody>
</table>

Where:
- **Data Out**: Tag Response returned by the reader.
- **SW1 SW2**: Status Code returned by the reader.

### Status Code

<table>
<thead>
<tr>
<th>Results</th>
<th>SW1</th>
<th>SW2</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>90</td>
<td>00h</td>
<td>The operation is completed successfully.</td>
</tr>
<tr>
<td>Error</td>
<td>63</td>
<td>00h</td>
<td>The operation is failed.</td>
</tr>
</tbody>
</table>

**Example 1:** To Write-with-erase (8 Bytes) a Topaz512/Jewel96 tag.

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be “FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF”

In which,
- **Direct Transmit APDU** = “FF 00 00 00”
- **Length of the Tag Command** = “0D”
- **Tag Command (InDataExchange)** = “D4 40 01”
- **Tag Command (Write-with-erase 8Bytes)** = “54”
- **Tag Address (00~3F (hex))** = “05”
- **Tag Data** = “01 23 45 67 89 AB CD EF”

To send an APDU to the slot 0 (default), sequence number = 1.

HOST > 02 6F 12 00 00 00 00 01 00 00 00
FF 00 00 00 0D D4 40 01 54 05 01 23 45 67 89 AB CD EF
[Checksum] 03

RDR -> 02 00 00 03

RDR -> 02 80 0D 00 00 00 00 01 01 00 00
D5 09 05 01 23 45 67 89 AB CD EF 90 00
[Checksum] 03

The APDU Response is “D5 09 05 01 23 45 67 89 AB CD EF 90 00”
In which,

Response returned by the contactless chip = “D5 09 05 01 23 45 67 89 AB CD EF 90 00”
Write Tag Address = “05”
Write Tag 8 Bytes Data = “01 23 45 67 89 AB CD EF”
Status Code returned by the reader = “90 00”

Example 2: To Write-no-erase (8 Bytes) a Topaz512/Jewel96 tag.

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be “FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF”

In which,

Direct Transmit APDU = “FF 00 00 00”
Length of the Tag Command = “0D”
Tag Command (InDataExchange) = “D4 40 01”
Tag Command (Write-no-erase 8Bytes) = “1B”
Tag Address (00~3F (hex)) = “05”
Tag Data = “FF FF FF FF FF FF FF FF”

To send an APDU to the slot 0 (default), sequence number = 1.

HOST -> 02 6F 12 00 00 00 01 00 00 00
      FF 00 00 00 0D D4 40 01 1B 05 FF FF FF FF FF FF FF FF
[Checksum] 03
RDR -> 02 00 00 03
RDR -> 02 80 0D 00 00 00 00 01 01 00 00
      D5 09 05 FF FF FF FF FF FF FF 90 00
[Checksum] 03

The APDU Response is “D5 09 05 FF FF FF FF FF FF FF 90 00”

In which,

Response returned by the contactless chip = “D5 09 05 FF FF FF FF FF FF FF 90 00”
Write Tag Address = “05”
Write Tag 8Bytes Data = “FF FF FF FF FF FF FF”
Status Code returned by the reader = “90 00”
Example 3: To Read 8 Bytes a Topaz512/Jewel96 Tag.

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be "FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00 00 00 00 00"

In which,
- Direct Transmit APDU = “FF 00 00 00”
- Length of the Tag Command = “0D”
- Tag Command (InDataExchange) = “D4 40 01”
- Tag Command (Read 8Bytes) = “02”
- Tag Address (00~3F (hex)) = “05”
- Tag Data = “00 00 00 00 00 00 00 00”

To send an APDU to the slot 0 (default), sequence number = 1.

HOST -> 02 6F 12 00 00 00 00 01 00 00 00
FF 00 00 00 0D D4 40 01 02 05 00 00 00 00 00 00 00 00 00
[Checksum] 03
RDR -> 02 00 00 03

RDR -> 02 80 0D 00 00 00 00 01 01 00 00
D5 09 05 01 23 45 67 89 AB CD EF 90 00
[Checksum] 03

The APDU Response is “D5 09 05 01 23 45 67 89 AB CD EF 90 00”

In which,
- Response returned by the contactless chip = “D5 09 05 01 23 45 67 89 AB CD EF 90 00”
- Read Tag Address = “05”
- Read Tag 8Bytes Data = “01 23 45 67 89 AB CD EF”
- Status Code returned by the reader = “90 00”

Example 4: To Read Segment a Topaz512/Jewel96 Tag.

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be “FF 00 00 00 0D D4 40 01 10 00 00 00 00 00 00 00 00 00 00 00 00 00”

In which,
- Direct Transmit APDU = “FF 00 00 00”
Length of the Tag Command = "0D"
Tag Command (InDataExchange) = "D4 40 01"
Tag Command (Read Segment) = "10"
Tag Address (00/10/20/30) = "00 " (Block 0)
Tag Data = "00 00 00 00 00 00 00 00"

To send an APDU to the slot 0 (default), sequence number = 1.

HOST -> 02 6F 12 00 00 00 01 00 00 00
       FF 00 00 00 00 D4 40 01 10 00 00 00 00 00 00 00 00 00 00 00
       [Checksum] 03

RDR -> 02 00 00 03
RDR -> 02 80 0D 00 00 00 00 01 01 00 00
       D5 41 00 ... <128 bytes data> ... 90 00
       [Checksum] 03

The APDU Response is "D5 41 00 ... <128 bytes data> ... 90 00"

In which,
   Response returned by the contactless chip = "D5 41 00 ... <128 bytes data> ... 90 00"
   Read Tag Segment Data = "<128 bytes data>"
   Status Code returned by the reader = "90 00"

Example 5: To Write Multi-Data at Topaz/Jewel Tag.

Note: This function only can write at the segment 0.

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be "FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09
       10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
       42 43 44 45 46 47"

In which,
   Direct Transmit APDU = "FF 00 00 00"
   Length of the Tag Command = "36"
   Tag Command (InDataExchange) = "D4 40 01"
   Tag Command (Write Multi-Data) = "58"

   Tag Address = "20 (0 0100 000) " (Block 4, Byte-0) (refer to Figure 2)

   Tag Data = "00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26"
To send an APDU to the slot 0 (default), sequence number = 1.

HOST -> 02 6F 3B 00 00 00 00 01 00 00 00
   FF 00 00 00 36 D4 40 01 58 20 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
   17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
   45 46 47
   [Checksum] 03

RDR -> 02 00 00 03
RDR -> 02 80 05 00 00 00 00 01 01 00 00
   D5 41 00 90 00
   [Checksum] 03

The APDU Response is “D5 41 00 90 00”

In which,

Response returned by the contactless chip = “D5 41 00 90 00”
Write Tag Data = “00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”
Status Code returned by the reader = “90 00”
If Status Code returned by the reader = “63 00” that means this operation is not complete.

Example 6: To Write Multi-8 bytes Data at Topaz512/Jewel96 tag.

Step 1. Issue a “Direct Transmit” APDU.

The APDU Command should be “FF 00 00 00 36 D4 40 01 5A 0 4 30 00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
42 43 44 45 46 47”

In which,

Direct Transmit APDU = “FF 00 00 00”
Length of the Tag Command = “36”
Tag Command (InDataExchange) = “D4 40 01”

Tag Command (Write Multi-Data) = “5A”
Tag Address (Block No.00-3F) = “04” (Block No. 4) (refer to below Fig. 3)

Tag Data = “00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”

![Address operand ‘ADD8’](image)

**Figure 3**: Tag Address “ADD8”

To send an APDU to the slot 0 (default), sequence number = 1.

**HOST ->**
02 6F 3B 00 00 00 00 01 00 00 00
FF 00 00 36 D4 40 01 5A 04 30 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47

[Checksum] 03

**RDR ->**
02 00 00 03

**RDR ->**
02 80 04 00 00 00 00 01 01 00 00

D5 09 90 00

[Checksum] 03

The APDU Response is “D5 09 90 00”

In which,

Response returned by the contactless chip = “D5 09 90 00”

Write Tag Data = “00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47”

Status Code returned by the reader = “90 00”

If Status Code returned by the reader = “63 00” that mean this operation is not complete.
6.6. Basic program flow for ISO 14443-4 Type A and B tags

Typical sequence may be:

1. Scanning the tags in the field (Polling) with the correct parameter (Type A or B).
2. Change the Baud Rate (optional for Type A tags only).
3. Perform any T=CL command.
4. Deselect the tag.

Step 1. Set the retry time.

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 04 00 00 00 00 01 01 00 00
RDR -> D5 33 90 00 [Checksum] 03

In which, Number of Tag found = [01];
Target number = 01
SENS_RES = 00 08;
SEL_RES = 28,
Length of the UID = 4;
UID = 85 82 2F A0
ATS = 07 77 F7 80 02 47 65
Operation Finished = 90 00

OR

Step 2. Polling for the ISO14443-4 Type A Tag, 106 kbps.

HOST -> 02 6F 09 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 08 28 04 85 82 2F A0 07 77 F7 80 02 47 65 90 00 [Checksum] 03

In which, Number of Tag found = [01];
Target number = 01
SENS_RES = 00 08; SEL_RES = 28,
Length of the UID = 4;
UID = 85 82 2F A0
ATS = 07 77 F7 80 02 47 65
Operation Finished = 90 00

OR

Step 2. Polling for the ISO14443-4 Type B Tag, 106 kbps.

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 05 D4 4A 01 03 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 14 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 50 00 01 32 F4 00 00 00 00 33 81 81 01 21 90 00 [Checksum] 03
In which,  
Number of Tag found = [01];
Target number = 01
ATQB = 50 00 01 32 F4 00 00 00 33 81 81
ATTRIB_RES Length = 01;   ATTRIB_RES = 21
Operation Finished = 90 00

Step 3. Change the default Baud Rate to other Baud Rate (optional).
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 05 D4 4E 01 02 02 [Checksum] 03 // Change to Baud Rate 424 kbps
Or
HOST -> FF 00 00 00 05 D4 4E 01 01 01 [Checksum] 03 // Change to Baud Rate 212 kbps
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 4F [00] 90 00 [Checksum] 03

Note: Please check the maximum baud rate supported by the tags. Only Type A tags are supported.

Step 4. Perform T=CL command, Get Challenge APDU = 00 84 00 00 08.
HOST -> 02 6F 0D 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 08 D4 40 01 00 84 00 00 08 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0F 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 62 89 99 ED C0 57 69 2B 90 00 90 00 [Checksum] 03
In which, Response Data = 62 89 99 ED C0 57 69 2B 90 00

Step 5. Deselect the tag.
HOST -> 02 6F 08 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Step 6. Turn off the Antenna Power (optional).
HOST -> 02 6F 09 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 04 D4 32 01 00
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 04 00 00 00 00 01 01 00 00
RDR -> D5 33 90 00 [Checksum] 03

Note: Please refer to the Tag specification for more detailed information.
6.7. Basic program flow for MIFARE applications

Typical sequence may be:

1. Scanning the tags in the field (Polling).
2. Authentication.
3. Read/Write the memory of the tag.
4. Halt the tag (optional).

Step 1. Set the retry time.
HOST -> 02 6F 0B 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 04 00 00 00 01 01 00 00
RDR -> D5 33 90 00 [Checksum] 03

Step 2. Polling for the MIFARE 1K/4K tags, 106 kbps.
HOST -> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 4A 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0E 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 02 18 04 F6 8E 2A 99 90 00 [Checksum] 03

In which, Number of Tag found = [01]; Target number = 01
SENS_RES = 00 02; SEL_RES = 18,
Length of the UID = 4;
UID = F6 8E 2A 99
Operation Finished = 90 00

Note: The tag type can be determined by recognizing the SEL_RES.

SEL_RES of some common tag types.
00 = MIFARE® Ultralight®
08 = MIFARE® Classic 1K
09 = MIFARE® Mini
18 = MIFARE® Classic 4K
20 = MIFARE® DESFire®
28 = JCOP30
98 = Gemplus MPCOS

Step 3. KEY A Authentication, Block 04, KEY = FF FF FF FF FF FF, UID = F6 8E 2A 99
HOST -> 02 6F 14 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 0F D4 40 01 60 04 FF FF FF FF FF FF F6 8E 2A 99 [Checksum] 03
RDR → 02 00 00 03 (Waiting the Tag)
RDR → 02 80 05 00 00 00 00 01 01 00 00
RDR → D5 41 [00] 90 00 [Checksum] 03

Note: If the authentication failed, the error code [XX] will be returned.
[00] = Valid, other = Error. Please refer to Error Codes Table for more details.

For KEY B Authentication
HOST → 02 6F 14 00 00 00 00 01 00 00 00
HOST → FF 00 00 00 0F D4 40 01 61 04 FF FF FF FF FF FF F6 8E 2A 99
RDR → 02 00 00 03 (Waiting the Tag)
RDR → 02 80 05 00 00 00 00 01 01 00 00
RDR → D5 41 [00] 90 00 [Checksum] 03

Step 4. Read the content of Block 04.
HOST → 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR → 02 00 00 03 (Waiting the Tag)
RDR → 02 80 15 00 00 00 00 01 01 00 00
RDR → D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03
In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Step 5. Update the content of Block 04.
HOST → 02 6F 1A 00 00 00 00 01 00 00 00
HOST → FF 00 00 00 15 D4 40 01 A0 04 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 [Checksum] 03
RDR → 02 00 00 03 (Waiting the Tag)
RDR → 02 80 05 00 00 00 00 01 01 00 00
RDR → D5 41 [00] 90 00 [Checksum] 03

Step 6. Halt the tag (optional).
HOST → 02 6F 08 00 00 00 00 01 00 00 00
HOST → FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR → 02 00 00 03 (Waiting the Tag)
RDR → 02 80 05 00 00 00 00 01 01 00 00
RDR → D5 45 [00] 90 00 [Checksum] 03
6.7.1. Handling the value blocks of MIFARE 1K/4K tag?

The value blocks are used for performing electronic purse functions (Increment, Decrement, Restore, Transfer, etc). These value blocks have a fixed data format which permits error detection and correction and a backup management.

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Value</td>
<td>_____</td>
<td>Value</td>
<td>ADR</td>
<td>_____</td>
<td>ADR</td>
<td>_____</td>
<td>ADR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where:

**Value**
A signed 4-byte value. The lowest significant byte off a value is stored in the lowest address byte. Negative values are stored in standard 2’s complement format.

**Adr**
1-Byte address, which can be used to save the storage address of a block. (optional).

**Example:**

Value 100 (decimal) = 64 (Hex), assume Block = 05h

The formatted value block = 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA

Step 1. Update the content of Block 05 with a value 100 (dec).

HOST -> 02 6F 1A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 01 A0 05 64 00 00 00 9B FF FF FF 64 00 00 00 05 FA 05 FA [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Step 2. Increment the value of Block 05 by 1 (dec).

HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 09 D4 40 01 C1 05 01 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

**Note: Decrement the value of Block 05 by 1 (dec).**

HOST -> FF 00 00 00 09 D4 40 01 C0 05 01 00 00 00

Step 3. Transfer the prior calculated value of Block 05 (dec).

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 B0 05 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

*Note:* Restore the value of Block 05 (cancel the prior Increment or Decrement operation).

HOST -> FF 00 00 00 05 D4 40 01 C2 05

Step 4. Read the content of Block 05.
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 05 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 65 00 00 00 9A FF FF 65 00 00 00 05 FA 05 FA 90 00 [Checksum] 03

In which, the value = 101 (dec)

Step 5. Copy the value of Block 05 to Block 06 (dec).
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 C2 05 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 B0 06 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Step 6. Read the content of Block 06.
HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 06 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 65 00 00 00 9A FF FF 65 00 00 00 05 FA 05 FA 90 00 [Checksum] 03

In which, the value = 101 (dec). The Adr "05 FA 05 FA" tells us the value is copied from Block 05.

*Note:* Please refer to the MIFARE specification for more detailed information.
### Table 2: MIFARE 1K Memory Map

<table>
<thead>
<tr>
<th>Sectors (Total 16 sectors. Each sector consists of 4 consecutive blocks)</th>
<th>Data Blocks (3 blocks, 16 bytes per block)</th>
<th>Trailer Block (1 block, 16 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sector 0</td>
<td>00h ~ 02h</td>
<td>03h</td>
</tr>
<tr>
<td>Sector 1</td>
<td>04h ~ 06h</td>
<td>07h</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>Sector 14</td>
<td>38h ~ 0Ah</td>
<td>3Bh</td>
</tr>
<tr>
<td>Sector 15</td>
<td>3Ch ~ 3Eh</td>
<td>3Fh</td>
</tr>
</tbody>
</table>

### Table 3: MIFARE 4K Memory Map

<table>
<thead>
<tr>
<th>Sectors (Total 8 sectors. Each sector consists of 16 consecutive blocks)</th>
<th>Data Blocks (15 blocks, 16 bytes per block)</th>
<th>Trailer Block (1 block, 16 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sector 32</td>
<td>80h ~ 8Eh</td>
<td>8Fh</td>
</tr>
<tr>
<td>Sector 33</td>
<td>90h ~ 9Eh</td>
<td>9Fh</td>
</tr>
<tr>
<td>..</td>
<td>..</td>
<td>..</td>
</tr>
<tr>
<td>Sector 38</td>
<td>E0h ~ EEh</td>
<td>EFh</td>
</tr>
<tr>
<td>Sector 39</td>
<td>F0h ~ FEh</td>
<td>FFh</td>
</tr>
</tbody>
</table>

**Note:** Once the authentication is done, all the data blocks of the same sector are free to access. For example, once the data block 04h is successfully authenticated (Sector 1), the data blocks 04h ~ 07h are free to access.
6.7.2. Accessing MIFARE Ultralight tags

Typical sequence may be:

1. Scanning the tags in the field (Polling).
2. Read/Write the memory of the tag.
3. Halt the tag (optional).

Step 1. Set the retry time.

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST -> FF 00 00 06 D4 32 05 00 00 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 04 00 00 00 00 01 01 00 00
RDR -> D5 33 90 00 [Checksum] 03

Step 2. Polling for the MIFARE Ultralight tags, 106 kbp.

HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 11 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00 [Checksum] 03

In which, Number of Tag found = [01];
Target number = 01
SENS_RES = 00 44; SEL_RES = 00,
Length of the UID = 7;
UID = 04 6E 0C A1 BF 02 84
Operation Finished = 90 00

Step 3. Read the content of Page 04.

HOST -> 02 6F 0A 00 00 00 00 01 01 00 00
HOST -> FF 00 00 00 05 D4 40 01 00 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 11 12 13 14 15 16 90 00 [Checksum] 03

In which, Block Data = 01 02 03 04 05 06 07 08 09 11 12 13 14 15 16

Note: 4 consecutive Pages will be retrieved. Pages 4, 5, 6 and 7 will be retrieved. Each data page consists of 4 bytes.
Step 4. Update the content of Page 04 with the data “AA BB CC DD”.

HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 09 D4 40 01 A2 04 AA BB CC DD [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

OR

Step 4. Write (MIFARE compatible Write) the content of Page 04 with the data “AA BB CC DD”.

HOST -> 02 6F 1A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 A0 04 AA BB CC DD 00 00 00 00 00 00 00 00 00 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

Note: This command is implemented to accommodate the established MIFARE 1K/4K infrastructure. We have to assemble the data into a 16 bytes frame. The first 4 bytes are for data, the rest of the bytes (12 ZEROS) are for padding. Only the page 4 (4 bytes) is updated even though the whole 16 bytes frame is sent to the reader.

Step 5. Read the content of Page 04 again.

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03

In which, Block Data = AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16

Note: Only the page 4 is updated. Pages 5, 6 and 7 remain the same.

Step 6. Halt the tag (optional).

HOST -> 02 6F 08 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 01 01 00 00
RDR -> D5 45 [00] 90 00 [Checksum] 03

Note: Please refer to the MIFARE Ultralight specification for more detailed information.
6.7.3. Accessing MIFARE Ultralight C tag

Typical sequence may be:

1. Scanning the tags in the field (Polling).
2. Authentication.
3. Read/Write the memory of the tag.
4. Halt the tag (optional).

Step 1. Set the retry time.

HOST → 02 6F 0B 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)
HOST → FF 00 00 00 04 D4 32 05 00 00 01 [Checksum] 03
RDR → 02 00 00 03 (Waiting the Tag)
RDR → 02 80 04 00 00 00 00 01 01 00 00
RDR → D5 33 90 00 [Checksum] 03

Step 2. Polling for the MIFARE Ultralight C tags, 106 kbps.

HOST → 02 6F 0B 00 00 00 00 01 00 00 00
HOST → FF 00 00 00 04 D4 4A 01 00 [Checksum] 03
RDR → 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 11 00 00 00 00 01 01 00 00
RDR -> D5 4B 01 01 00 44 00 07 04 6E 0C A1 BF 02 84 90 00 [Checksum] 03

In which,  Number of Tag found = [01]; Target number = 01
SENS_RES = 00 44; SEL_RES = 00,
Length of the UID = 7;
UID = 04 6E 0C A1 BF 02 84
Operation Finished = 90 00

Step 3. 3DES Authentication.
HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 04 D4 42 1A 00 10 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0E 00 00 00 00 01 01 00 00
RDR -> D5 43 [00] 04 77 64 89 99 74 24 67 90 00 [Checksum] 03

In which, 3DES challenge from the card = [04 77 64 89 99 74 24 67];
h = 90 00
HOST -> 02 6F 18 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 13 D4 42 AF 88 68 45 07 65 86 99 67 00 53 77 56 98 65 49 67 [Checksum] 03

In which, 3DES reply to the card = [88 68 45 07 65 86 99 67 00 53 77 56 98 65 49 67];
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 0E 00 00 00 00 01 01 00 00
RDR -> D5 43 [00] 00 06 78 53 80 68 89 61 24 90 00 [Checksum] 03

In which, 3DES reply from the card = [06 78 53 80 68 89 61 24];
Operation Finished = 90 00

Note: The 3DES reply from the card should be checked to make sure the card is legitimate.

Step 4. Read the content of Page 04.
HOST -> 02 6F 09 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 00 01 01 00 00
RDR -> D5 41 [00] 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03
In which, Block Data = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

**Note:** 4 consecutive Pages will be retrieved. Pages 4, 5, 6 and 7 will be retrieved. Each data page consists of 4 bytes.

Step 5. Update the content of Page 04 with the data “AA BB CC DD”.

HOST -> 02 6F 0E 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 09 D4 40 01 A2 04 AA BB CC DD [Checksum] 03
RDR -> 02 80 05 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

OR

Step 5. Write (MIFARE compatible Write) the content of Page 04 with the data “AA BB CC DD”.

HOST -> 02 6F 1A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 15 D4 40 01 A0 04 AA BB CC DD 00 00 00 00 00 00 00 00 00 00 00 00 00 00 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 01 01 00 00
RDR -> D5 41 [00] 90 00 [Checksum] 03

**Note:** This command is implemented to accommodate the established MIFARE 1K/4K infrastructure. We have to assemble the data into a 16 bytes frame. The first 4 bytes are for data, the rest of the bytes (12 ZEROS) are for padding. Only the page 4 (4 bytes) is updated even though the whole 16 bytes frame is sent to the reader.

Step 6. Read the content of Page 04 again.

HOST -> 02 6F 0A 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 05 D4 40 01 30 04 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 15 00 00 00 01 01 00 00
RDR -> D5 41 [00] AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16 90 00 [Checksum] 03

In which, Block Data = AA BB CC DD 05 06 07 08 09 10 11 12 13 14 15 16

**Note:** Only the page 4 is updated. Pages 5, 6 and 7 remain the same.

Step 7. Halt the tag (optional).

HOST -> 02 6F 08 00 00 00 00 01 00 00 00
HOST -> FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR -> 02 00 00 03 (Waiting the Tag)
RDR -> 02 80 05 00 00 00 01 01 00 00
RDR -> D5 45 [00] 90 00 [Checksum] 03

*Note:* Please refer to the MIFARE Ultralight C specification for more detailed information.

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial Number</td>
<td>SN0</td>
<td>SN1</td>
<td>SN2</td>
<td>BCC0</td>
<td>0</td>
</tr>
<tr>
<td>Serial Number</td>
<td>SN3</td>
<td>SN4</td>
<td>SN5</td>
<td>SN6</td>
<td>1</td>
</tr>
<tr>
<td>Internal/Lock</td>
<td>BCC1</td>
<td>Internal</td>
<td>Lock</td>
<td>Lock</td>
<td>2</td>
</tr>
<tr>
<td>OTP</td>
<td>OTP0</td>
<td>OTP1</td>
<td>OTP2</td>
<td>OTP3</td>
<td>3</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data0</td>
<td>Data1</td>
<td>Data2</td>
<td>Data3</td>
<td>4</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data4</td>
<td>Data5</td>
<td>Data6</td>
<td>Data7</td>
<td>5</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data8</td>
<td>Data9</td>
<td>Data10</td>
<td>Data11</td>
<td>6</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data12</td>
<td>Data13</td>
<td>Data14</td>
<td>Data15</td>
<td>7</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data16</td>
<td>Data17</td>
<td>Data18</td>
<td>Data19</td>
<td>8</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data20</td>
<td>Data21</td>
<td>Data22</td>
<td>Data23</td>
<td>9</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data24</td>
<td>Data25</td>
<td>Data26</td>
<td>Data27</td>
<td>10</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data28</td>
<td>Data29</td>
<td>Data30</td>
<td>Data31</td>
<td>11</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data32</td>
<td>Data33</td>
<td>Data34</td>
<td>Data35</td>
<td>12</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data36</td>
<td>Data37</td>
<td>Data38</td>
<td>Data39</td>
<td>13</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data40</td>
<td>Data41</td>
<td>Data42</td>
<td>Data43</td>
<td>14</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data44</td>
<td>Data45</td>
<td>Data46</td>
<td>Data47</td>
<td>15</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data48</td>
<td>Data49</td>
<td>Data50</td>
<td>Data51</td>
<td>16</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data52</td>
<td>Data53</td>
<td>Data54</td>
<td>Data55</td>
<td>17</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data56</td>
<td>Data57</td>
<td>Data58</td>
<td>Data59</td>
<td>18</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data60</td>
<td>Data61</td>
<td>Data62</td>
<td>Data63</td>
<td>19</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data64</td>
<td>Data65</td>
<td>Data66</td>
<td>Data67</td>
<td>20</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data68</td>
<td>Data69</td>
<td>Data70</td>
<td>Data71</td>
<td>21</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data72</td>
<td>Data73</td>
<td>Data74</td>
<td>Data75</td>
<td>22</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data76</td>
<td>Data77</td>
<td>Data78</td>
<td>Data79</td>
<td>23</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data80</td>
<td>Data81</td>
<td>Data82</td>
<td>Data83</td>
<td>24</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data84</td>
<td>Data85</td>
<td>Data86</td>
<td>Data87</td>
<td>25</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data88</td>
<td>Data89</td>
<td>Data90</td>
<td>Data91</td>
<td>26</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data92</td>
<td>Data93</td>
<td>Data94</td>
<td>Data95</td>
<td>27</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data96</td>
<td>Data97</td>
<td>Data98</td>
<td>Data99</td>
<td>28</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data100</td>
<td>Data101</td>
<td>Data102</td>
<td>Data103</td>
<td>29</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data104</td>
<td>Data105</td>
<td>Data106</td>
<td>Data107</td>
<td>30</td>
</tr>
<tr>
<td>Byte Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data108</td>
<td>Data109</td>
<td>Data110</td>
<td>Data111</td>
<td>31</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data112</td>
<td>Data113</td>
<td>Data114</td>
<td>Data115</td>
<td>32</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data116</td>
<td>Data117</td>
<td>Data118</td>
<td>Data119</td>
<td>33</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data120</td>
<td>Data121</td>
<td>Data122</td>
<td>Data123</td>
<td>34</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data124</td>
<td>Data125</td>
<td>Data126</td>
<td>Data127</td>
<td>35</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data128</td>
<td>Data129</td>
<td>Data130</td>
<td>Data131</td>
<td>36</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data132</td>
<td>Data133</td>
<td>Data134</td>
<td>Data135</td>
<td>37</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data136</td>
<td>Data137</td>
<td>Data138</td>
<td>Data139</td>
<td>38</td>
</tr>
<tr>
<td>Data read/write</td>
<td>Data140</td>
<td>Data141</td>
<td>Data142</td>
<td>Data143</td>
<td>39</td>
</tr>
<tr>
<td>Lock</td>
<td>Lock</td>
<td>Lock</td>
<td>-</td>
<td>-</td>
<td>40</td>
</tr>
<tr>
<td>16 bit counter</td>
<td>16 bit counter</td>
<td>16 bit counter</td>
<td>-</td>
<td>-</td>
<td>41</td>
</tr>
<tr>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>42</td>
</tr>
<tr>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>Authentication configuration</td>
<td>43</td>
</tr>
<tr>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>44</td>
</tr>
<tr>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>45</td>
</tr>
<tr>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>46</td>
</tr>
<tr>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>Authentication key</td>
<td>47</td>
</tr>
</tbody>
</table>

**Table 5: MIFARE Ultralight C Memory Map**

**Total Page Size:** 792 bits of 198 Bytes.
### 6.8. Basic program flow for FeliCa applications

**Step 0.** Start the application. The first thing to do is to activate the “SAM Interface”. The ATR of the SAM (if a SAM is inserted) or a Pseudo-ATR “3B 00” (if no SAM is inserted) will be returned. In other words, the SAM always exists from the view of the application.

**Step 1.** The second thing to do is to change the operating parameters of the contactless chip. Set the Retry Time to two.

**Step 2.** Poll a FeliCa Tag by sending “Direct Transmit” and “Get Response” APDUs (Tag Polling).

**Step 3.** If no tag is found, go back to Step 2 until a FeliCa Tag is found.

**Step 4.** Access the FeliCa Tag by sending APDUs (Tag Read or Write).

**Step 5.** If there is no any operation with the FeliCa Tag, then go back to Step 2 to poll the other FeliCa Tag.

**..**

**Step N.** Deactivate the “SAM Interface”. Shut down the application.

**Notes:**

1. The default Retry Time of the Tag command “InListPassiveTarget” is infinity. Send the APDU “FF 00 00 00 06 D4 32 05 00 00 01” to change the Retry Time to two.

2. It is recommended to turn off the Antenna if there is no contactless access.

   APDU for turning on the Antenna Power = APDU “FF 00 00 00 04 D4 32 01 03”

   APDU for turning off the Antenna Power = APDU “FF 00 00 00 04 D4 32 01 02”

### 6.9. Basic program flow for NFC Forum Type 1 tag applications

**Example:** Jewel and Topaz tags.

Typical sequence may be:

1. Scanning the tags in the field (Polling).
2. Read/Update the memory of the tag.
3. Deselect the tag.

**Step 1.** Set the retry time.

HOST -> 02 6F 0B 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)

HOST -> FF 00 00 00 06 D4 32 05 00 00 01 [Checksum] 03

RDR -> 02 00 00 03 (Waiting the Tag)

RDR -> D5 33 90 00 [Checksum] 03

**Step 2.** Polling for the Jewel or Topaz Tag, 106 kbps.

HOST -> 02 6F 09 00 00 00 00 01 00 00 00 (HOST_to_RDR_XfrBlock Format)

HOST -> FF 00 00 00 04 D4 4A 01 04 [Checksum] 03

RDR -> 02 00 00 03 (Waiting the Tag)

RDR -> D5 4B 01 01 0C 00 B5 3E 21 00 90 00 [Checksum] 03
In which, Number of Tag found = [01];
Target number = 01
ATQA_RES = 0C 00;
UID = B5 3E 21 00
Operation Finished = 90 00

Step 3. Read the memory address 08 (Block 1: Byte-0)
HOST  -> 02 6F 0A 00 00 00 00 01 00 00 00 FF 00 00 00 05 D4 40 01 01 08 [Checksum] 03
RDR   -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 18 90 00 [Checksum] 03
In which, Response Data = 18

Note: To read all the memory content of the tag starting from the memory address 00:
HOST  -> 02 6F 09 00 00 00 00 01 00 00 00 FF 00 00 00 04 D4 40 01 00 [Checksum] 03
RDR   -> 02 00 00 03 02 80 7F 00 00 00 00 01 01 00 00 D5 41 00 11 48
RDR   -> show all data … 90 00 [Checksum] 03

Step 4. Update the memory address 08 (Block 1: Byte-0) with the data FF
HOST  -> 2 6F 0B 00 00 00 00 01 00 00 00 FF 00 00 00 06 D4 40 01 53 08 FF [Checksum] 03
RDR   -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 41 [00] FF 90 00 [Checksum] 03
In which, Response Data = FF

Note: To update more than one memory content of the tag starting from the memory address 08 (Block 1: Byte-0)
HOST  -> 02 6F 0D 00 00 00 00 01 00 00 00 FF 00 00 00 08 D4 40 01 58 08 02 AA BB [Checksum] 03
RDR   -> 02 00 00 03 02 80 06 00 00 00 00 01 01 00 00 D5 41 [00] 90 00 [Checksum] 03
In which, Command = 58;
Starting memory address = 08;
Number of write content = 02;
Memory content = AA, BB;

Step 5. Deselect the tag.
HOST  -> 02 6F 08 00 00 00 00 01 00 00 00 FF 00 00 00 03 D4 44 01 [Checksum] 03
RDR   -> 02 00 00 03 02 80 05 00 00 00 00 01 01 00 00 D5 45 [00] 90 00 [Checksum] 03
## Appendix A. Topaz

### EEPROM Memory Map

<table>
<thead>
<tr>
<th>Type</th>
<th>Block No.</th>
<th>Byte-0 (LSB)</th>
<th>Byte-1</th>
<th>Byte-2</th>
<th>Byte-3</th>
<th>Byte-4</th>
<th>Byte-5</th>
<th>Byte-6</th>
<th>Byte-7 (MSB)</th>
<th>Lockable</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>0</td>
<td>UID-0</td>
<td>UID-1</td>
<td>UID-2</td>
<td>UID-3</td>
<td>UID-4</td>
<td>UID-5</td>
<td>UID-6</td>
<td>Locked</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>1</td>
<td>Data0</td>
<td>Data1</td>
<td>Data2</td>
<td>Data3</td>
<td>Data4</td>
<td>Data5</td>
<td>Data6</td>
<td>Data7</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2</td>
<td>Data8</td>
<td>Data9</td>
<td>Data10</td>
<td>Data11</td>
<td>Data12</td>
<td>Data13</td>
<td>Data14</td>
<td>Data15</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3</td>
<td>Data16</td>
<td>Data17</td>
<td>Data18</td>
<td>Data19</td>
<td>Data20</td>
<td>Data21</td>
<td>Data22</td>
<td>Data23</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>4</td>
<td>Data24</td>
<td>Data25</td>
<td>Data26</td>
<td>Data27</td>
<td>Data28</td>
<td>Data29</td>
<td>Data30</td>
<td>Data31</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>5</td>
<td>Data32</td>
<td>Data33</td>
<td>Data34</td>
<td>Data35</td>
<td>Data36</td>
<td>Data37</td>
<td>Data38</td>
<td>Data39</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>6</td>
<td>Data40</td>
<td>Data41</td>
<td>Data42</td>
<td>Data43</td>
<td>Data44</td>
<td>Data45</td>
<td>Data46</td>
<td>Data47</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>7</td>
<td>Data48</td>
<td>Data49</td>
<td>Data50</td>
<td>Data51</td>
<td>Data52</td>
<td>Data53</td>
<td>Data54</td>
<td>Data55</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>8</td>
<td>Data56</td>
<td>Data57</td>
<td>Data58</td>
<td>Data59</td>
<td>Data60</td>
<td>Data61</td>
<td>Data62</td>
<td>Data63</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>9</td>
<td>Data64</td>
<td>Data65</td>
<td>Data66</td>
<td>Data67</td>
<td>Data68</td>
<td>Data69</td>
<td>Data70</td>
<td>Data71</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>A</td>
<td>Data72</td>
<td>Data73</td>
<td>Data74</td>
<td>Data75</td>
<td>Data76</td>
<td>Data77</td>
<td>Data78</td>
<td>Data79</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>B</td>
<td>Data80</td>
<td>Data81</td>
<td>Data82</td>
<td>Data83</td>
<td>Data84</td>
<td>Data85</td>
<td>Data86</td>
<td>Data87</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>C</td>
<td>Data88</td>
<td>Data89</td>
<td>Data90</td>
<td>Data91</td>
<td>Data92</td>
<td>Data93</td>
<td>Data94</td>
<td>Data95</td>
<td>Yes</td>
</tr>
<tr>
<td>Reserved</td>
<td>D</td>
<td>LOCK-0</td>
<td>LOCK-1</td>
<td>OTP-0</td>
<td>OTP-1</td>
<td>OTP-2</td>
<td>OTP-3</td>
<td>OTP-4</td>
<td>OTP-5</td>
<td></td>
</tr>
</tbody>
</table>

- **Reserved for internal use**
- **User Block Lock & Status**
- **OTP bits**
### Appendix B. Topaz512

#### EEPROM Memory Map (Segment 0)

<table>
<thead>
<tr>
<th>Type</th>
<th>Block No.</th>
<th>Byte 0 (LSB)</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6</th>
<th>Byte 7 (MSB)</th>
<th>Lockable</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>00</td>
<td>UID-0</td>
<td>UID-1</td>
<td>UID-2</td>
<td>UID-3</td>
<td>UID-4</td>
<td>UID-5</td>
<td>2%</td>
<td>Locked</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>01</td>
<td>Data0</td>
<td>Data1</td>
<td>Data2</td>
<td>Data3</td>
<td>Data4</td>
<td>Data5</td>
<td>Data6</td>
<td>Data7</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>02</td>
<td>Data8</td>
<td>Data9</td>
<td>Data10</td>
<td>Data11</td>
<td>Data12</td>
<td>Data13</td>
<td>Data14</td>
<td>Data15</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>03</td>
<td>Data16</td>
<td>Data17</td>
<td>Data18</td>
<td>Data19</td>
<td>Data20</td>
<td>Data21</td>
<td>Data22</td>
<td>Data23</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>04</td>
<td>Data24</td>
<td>Data25</td>
<td>Data26</td>
<td>Data27</td>
<td>Data28</td>
<td>Data29</td>
<td>Data30</td>
<td>Data31</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>05</td>
<td>Data32</td>
<td>Data33</td>
<td>Data34</td>
<td>Data35</td>
<td>Data36</td>
<td>Data37</td>
<td>Data38</td>
<td>Data39</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>06</td>
<td>Data40</td>
<td>Data41</td>
<td>Data42</td>
<td>Data43</td>
<td>Data44</td>
<td>Data45</td>
<td>Data46</td>
<td>Data47</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>07</td>
<td>Data48</td>
<td>Data49</td>
<td>Data50</td>
<td>Data51</td>
<td>Data52</td>
<td>Data53</td>
<td>Data54</td>
<td>Data55</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>08</td>
<td>Data56</td>
<td>Data57</td>
<td>Data58</td>
<td>Data59</td>
<td>Data60</td>
<td>Data61</td>
<td>Data62</td>
<td>Data63</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>09</td>
<td>Data64</td>
<td>Data65</td>
<td>Data66</td>
<td>Data67</td>
<td>Data68</td>
<td>Data69</td>
<td>Data70</td>
<td>Data71</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>0A</td>
<td>Data72</td>
<td>Data73</td>
<td>Data74</td>
<td>Data75</td>
<td>Data76</td>
<td>Data77</td>
<td>Data78</td>
<td>Data79</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>0B</td>
<td>Data80</td>
<td>Data81</td>
<td>Data82</td>
<td>Data83</td>
<td>Data84</td>
<td>Data85</td>
<td>Data86</td>
<td>Data87</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>0C</td>
<td>Data88</td>
<td>Data89</td>
<td>Data90</td>
<td>Data91</td>
<td>Data92</td>
<td>Data93</td>
<td>Data94</td>
<td>Data95</td>
<td>Yes</td>
</tr>
<tr>
<td>Reserved</td>
<td>0D</td>
<td>LOCK-0</td>
<td>LOCK-1</td>
<td>OTP-0</td>
<td>OTP-1</td>
<td>OTP-2</td>
<td>OTP-3</td>
<td>OTP-4</td>
<td>OTP-5</td>
<td>NA</td>
</tr>
<tr>
<td>Lock/OTP</td>
<td>0E</td>
<td>OTP-6</td>
<td>OTP-7</td>
<td>LOCK-2</td>
<td>LOCK-3</td>
<td>LOCK-4</td>
<td>LOCK-5</td>
<td>LOCK-6</td>
<td>LOCK-7</td>
<td>NA</td>
</tr>
</tbody>
</table>

#### EEPROM Memory Map (Segment 1)

<table>
<thead>
<tr>
<th>Type</th>
<th>Block No.</th>
<th>Byte 0 (LSB)</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
<th>Byte 5</th>
<th>Byte 6</th>
<th>Byte 7 (MSB)</th>
<th>Lockable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>10</td>
<td>Data96</td>
<td>Data97</td>
<td>Data98</td>
<td>Data99</td>
<td>Data100</td>
<td>Data101</td>
<td>Data102</td>
<td>Data103</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>11</td>
<td>Data104</td>
<td>Data105</td>
<td>Data106</td>
<td>Data107</td>
<td>Data108</td>
<td>Data109</td>
<td>Data110</td>
<td>Data111</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>12</td>
<td>Data112</td>
<td>Data113</td>
<td>Data114</td>
<td>Data115</td>
<td>Data116</td>
<td>Data117</td>
<td>Data118</td>
<td>Data119</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>13</td>
<td>Data120</td>
<td>Data121</td>
<td>Data122</td>
<td>Data123</td>
<td>Data124</td>
<td>Data125</td>
<td>Data126</td>
<td>Data127</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>14</td>
<td>Data128</td>
<td>Data129</td>
<td>Data130</td>
<td>Data131</td>
<td>Data132</td>
<td>Data133</td>
<td>Data134</td>
<td>Data135</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>15</td>
<td>Data136</td>
<td>Data137</td>
<td>Data138</td>
<td>Data139</td>
<td>Data140</td>
<td>Data141</td>
<td>Data142</td>
<td>Data143</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>16</td>
<td>Data144</td>
<td>Data145</td>
<td>Data146</td>
<td>Data147</td>
<td>Data148</td>
<td>Data149</td>
<td>Data150</td>
<td>Data151</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>17</td>
<td>Data152</td>
<td>Data153</td>
<td>Data154</td>
<td>Data155</td>
<td>Data156</td>
<td>Data157</td>
<td>Data158</td>
<td>Data159</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>18</td>
<td>Data160</td>
<td>Data161</td>
<td>Data162</td>
<td>Data163</td>
<td>Data164</td>
<td>Data165</td>
<td>Data166</td>
<td>Data167</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>19</td>
<td>Data168</td>
<td>Data169</td>
<td>Data170</td>
<td>Data171</td>
<td>Data172</td>
<td>Data173</td>
<td>Data174</td>
<td>Data175</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>1A</td>
<td>Data176</td>
<td>Data177</td>
<td>Data178</td>
<td>Data179</td>
<td>Data180</td>
<td>Data181</td>
<td>Data182</td>
<td>Data183</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>1B</td>
<td>Data184</td>
<td>Data185</td>
<td>Data186</td>
<td>Data187</td>
<td>Data188</td>
<td>Data189</td>
<td>Data190</td>
<td>Data191</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>1C</td>
<td>Data192</td>
<td>Data193</td>
<td>Data194</td>
<td>Data195</td>
<td>Data196</td>
<td>Data197</td>
<td>Data198</td>
<td>Data199</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>1D</td>
<td>Data200</td>
<td>Data201</td>
<td>Data202</td>
<td>Data203</td>
<td>Data204</td>
<td>Data205</td>
<td>Data206</td>
<td>Data207</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>1E</td>
<td>Data208</td>
<td>Data209</td>
<td>Data210</td>
<td>Data211</td>
<td>Data212</td>
<td>Data213</td>
<td>Data214</td>
<td>Data215</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>1F</td>
<td>Data216</td>
<td>Data217</td>
<td>Data218</td>
<td>Data219</td>
<td>Data220</td>
<td>Data221</td>
<td>Data222</td>
<td>Data223</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### EEPROM Memory Map (Segment 12)

<table>
<thead>
<tr>
<th>Type</th>
<th>Block No.</th>
<th>Byte-0 (LSB)</th>
<th>Byte-1</th>
<th>Byte-2</th>
<th>Byte-3</th>
<th>Byte-4</th>
<th>Byte-5</th>
<th>Byte-6</th>
<th>Byte-7 (MSB)</th>
<th>Lockable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>20</td>
<td>Data224</td>
<td>Data225</td>
<td>Data226</td>
<td>Data227</td>
<td>Data228</td>
<td>Data229</td>
<td>Data230</td>
<td>Data231</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>21</td>
<td>Data232</td>
<td>Data233</td>
<td>Data234</td>
<td>Data235</td>
<td>Data236</td>
<td>Data237</td>
<td>Data238</td>
<td>Data239</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>22</td>
<td>Data240</td>
<td>Data241</td>
<td>Data242</td>
<td>Data243</td>
<td>Data244</td>
<td>Data245</td>
<td>Data246</td>
<td>Data247</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>23</td>
<td>Data248</td>
<td>Data249</td>
<td>Data250</td>
<td>Data251</td>
<td>Data252</td>
<td>Data253</td>
<td>Data254</td>
<td>Data255</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>24</td>
<td>Data256</td>
<td>Data257</td>
<td>Data258</td>
<td>Data259</td>
<td>Data260</td>
<td>Data261</td>
<td>Data262</td>
<td>Data263</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>25</td>
<td>Data264</td>
<td>Data265</td>
<td>Data266</td>
<td>Data267</td>
<td>Data268</td>
<td>Data269</td>
<td>Data270</td>
<td>Data271</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>26</td>
<td>Data272</td>
<td>Data273</td>
<td>Data274</td>
<td>Data275</td>
<td>Data276</td>
<td>Data277</td>
<td>Data278</td>
<td>Data279</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>27</td>
<td>Data280</td>
<td>Data281</td>
<td>Data282</td>
<td>Data283</td>
<td>Data284</td>
<td>Data285</td>
<td>Data286</td>
<td>Data287</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>28</td>
<td>Data288</td>
<td>Data289</td>
<td>Data290</td>
<td>Data291</td>
<td>Data292</td>
<td>Data293</td>
<td>Data294</td>
<td>Data295</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>29</td>
<td>Data296</td>
<td>Data297</td>
<td>Data298</td>
<td>Data299</td>
<td>Data300</td>
<td>Data301</td>
<td>Data302</td>
<td>Data303</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2A</td>
<td>Data304</td>
<td>Data305</td>
<td>Data306</td>
<td>Data307</td>
<td>Data308</td>
<td>Data309</td>
<td>Data310</td>
<td>Data311</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2B</td>
<td>Data312</td>
<td>Data313</td>
<td>Data314</td>
<td>Data315</td>
<td>Data316</td>
<td>Data317</td>
<td>Data318</td>
<td>Data319</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2C</td>
<td>Data320</td>
<td>Data321</td>
<td>Data322</td>
<td>Data323</td>
<td>Data324</td>
<td>Data325</td>
<td>Data326</td>
<td>Data327</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2D</td>
<td>Data328</td>
<td>Data329</td>
<td>Data330</td>
<td>Data331</td>
<td>Data332</td>
<td>Data333</td>
<td>Data334</td>
<td>Data335</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2E</td>
<td>Data336</td>
<td>Data337</td>
<td>Data338</td>
<td>Data339</td>
<td>Data340</td>
<td>Data341</td>
<td>Data342</td>
<td>Data343</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2F</td>
<td>Data344</td>
<td>Data345</td>
<td>Data346</td>
<td>Data347</td>
<td>Data348</td>
<td>Data349</td>
<td>Data350</td>
<td>Data351</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### EEPROM Memory Map (Segment 13)

<table>
<thead>
<tr>
<th>Type</th>
<th>Block No.</th>
<th>Byte-0 (LSB)</th>
<th>Byte-1</th>
<th>Byte-2</th>
<th>Byte-3</th>
<th>Byte-4</th>
<th>Byte-5</th>
<th>Byte-6</th>
<th>Byte-7 (MSB)</th>
<th>Lockable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>30</td>
<td>Data352</td>
<td>Data353</td>
<td>Data354</td>
<td>Data355</td>
<td>Data356</td>
<td>Data357</td>
<td>Data358</td>
<td>Data359</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>31</td>
<td>Data360</td>
<td>Data361</td>
<td>Data362</td>
<td>Data363</td>
<td>Data364</td>
<td>Data365</td>
<td>Data366</td>
<td>Data367</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>32</td>
<td>Data368</td>
<td>Data369</td>
<td>Data370</td>
<td>Data371</td>
<td>Data372</td>
<td>Data373</td>
<td>Data374</td>
<td>Data375</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>33</td>
<td>Data376</td>
<td>Data377</td>
<td>Data378</td>
<td>Data379</td>
<td>Data380</td>
<td>Data381</td>
<td>Data382</td>
<td>Data383</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>34</td>
<td>Data384</td>
<td>Data385</td>
<td>Data386</td>
<td>Data387</td>
<td>Data388</td>
<td>Data389</td>
<td>Data390</td>
<td>Data391</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>35</td>
<td>Data392</td>
<td>Data393</td>
<td>Data394</td>
<td>Data395</td>
<td>Data396</td>
<td>Data397</td>
<td>Data398</td>
<td>Data399</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>36</td>
<td>Data400</td>
<td>Data401</td>
<td>Data402</td>
<td>Data403</td>
<td>Data404</td>
<td>Data405</td>
<td>Data406</td>
<td>Data407</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>37</td>
<td>Data408</td>
<td>Data409</td>
<td>Data410</td>
<td>Data411</td>
<td>Data412</td>
<td>Data413</td>
<td>Data414</td>
<td>Data415</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>38</td>
<td>Data416</td>
<td>Data417</td>
<td>Data418</td>
<td>Data419</td>
<td>Data420</td>
<td>Data421</td>
<td>Data422</td>
<td>Data423</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>39</td>
<td>Data424</td>
<td>Data425</td>
<td>Data426</td>
<td>Data427</td>
<td>Data428</td>
<td>Data429</td>
<td>Data430</td>
<td>Data431</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3A</td>
<td>Data432</td>
<td>Data433</td>
<td>Data434</td>
<td>Data435</td>
<td>Data436</td>
<td>Data437</td>
<td>Data438</td>
<td>Data439</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3B</td>
<td>Data440</td>
<td>Data441</td>
<td>Data442</td>
<td>Data443</td>
<td>Data444</td>
<td>Data445</td>
<td>Data446</td>
<td>Data447</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3C</td>
<td>Data448</td>
<td>Data449</td>
<td>Data450</td>
<td>Data451</td>
<td>Data452</td>
<td>Data453</td>
<td>Data454</td>
<td>Data455</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3D</td>
<td>Data456</td>
<td>Data457</td>
<td>Data458</td>
<td>Data459</td>
<td>Data460</td>
<td>Data461</td>
<td>Data462</td>
<td>Data463</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3E</td>
<td>Data464</td>
<td>Data465</td>
<td>Data466</td>
<td>Data467</td>
<td>Data468</td>
<td>Data469</td>
<td>Data470</td>
<td>Data471</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3F</td>
<td>Data472</td>
<td>Data473</td>
<td>Data474</td>
<td>Data475</td>
<td>Data476</td>
<td>Data477</td>
<td>Data478</td>
<td>Data479</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Reserved for internal use
User Block Lock & Status
OTP bits
Appendix C. Jewel64

<table>
<thead>
<tr>
<th>Type</th>
<th>Block No.</th>
<th>Byte-0 (LSB)</th>
<th>Byte-1</th>
<th>Byte-2</th>
<th>Byte-3</th>
<th>Byte-4</th>
<th>Byte-5</th>
<th>Byte-6</th>
<th>Byte-7 (MSB)</th>
<th>Lockable</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>0</td>
<td>UID-0</td>
<td>UID-1</td>
<td>UID-2</td>
<td>UID-3</td>
<td>UID-4</td>
<td>UID-5</td>
<td>25%</td>
<td></td>
<td>Locked</td>
</tr>
<tr>
<td>Data</td>
<td>1</td>
<td>Data0</td>
<td>Data1</td>
<td>Data2</td>
<td>Data3</td>
<td>Data4</td>
<td>Data5</td>
<td>Data6</td>
<td>Data7</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2</td>
<td>Data8</td>
<td>Data9</td>
<td>Data10</td>
<td>Data11</td>
<td>Data12</td>
<td>Data13</td>
<td>Data14</td>
<td>Data15</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3</td>
<td>Data16</td>
<td>Data17</td>
<td>Data18</td>
<td>Data19</td>
<td>Data20</td>
<td>Data21</td>
<td>Data22</td>
<td>Data23</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>4</td>
<td>Data24</td>
<td>Data25</td>
<td>Data26</td>
<td>Data27</td>
<td>Data28</td>
<td>Data29</td>
<td>Data30</td>
<td>Data31</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>5</td>
<td>Data32</td>
<td>Data33</td>
<td>Data34</td>
<td>Data35</td>
<td>Data36</td>
<td>Data37</td>
<td>Data38</td>
<td>Data39</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>6</td>
<td>Data40</td>
<td>Data41</td>
<td>Data42</td>
<td>Data43</td>
<td>Data44</td>
<td>Data45</td>
<td>Data46</td>
<td>Data47</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>7</td>
<td>Data48</td>
<td>Data49</td>
<td>Data50</td>
<td>Data51</td>
<td>Data52</td>
<td>Data53</td>
<td>Data54</td>
<td>Data55</td>
<td>Yes</td>
</tr>
<tr>
<td>Reserved</td>
<td>8</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>NA</td>
</tr>
<tr>
<td>Reserved</td>
<td>9</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>NA</td>
</tr>
<tr>
<td>Reserved</td>
<td>A</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>NA</td>
</tr>
<tr>
<td>Reserved</td>
<td>B</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>NA</td>
</tr>
<tr>
<td>Data</td>
<td>C</td>
<td>Data56</td>
<td>Data57</td>
<td>Data58</td>
<td>Data59</td>
<td>Data60</td>
<td>Data61</td>
<td>Data62</td>
<td>Data63</td>
<td>Yes</td>
</tr>
<tr>
<td>Reserved</td>
<td>D</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>NA</td>
</tr>
<tr>
<td>Lock/OTP</td>
<td>E</td>
<td>LOCK-0</td>
<td>LOCK-1</td>
<td>OTP-0</td>
<td>OTP-1</td>
<td>OTP-2</td>
<td>OTP-3</td>
<td>OTP-4</td>
<td>OTP-5</td>
<td>NA</td>
</tr>
</tbody>
</table>

Reserved for internal use
User Block Lock & Status
OTP bits
## Appendix D. Jewel96

### EEPROM Memory Map

<table>
<thead>
<tr>
<th>Type</th>
<th>Block No</th>
<th>Byte-0 (LSB)</th>
<th>Byte-1</th>
<th>Byte-2</th>
<th>Byte-3</th>
<th>Byte-4</th>
<th>Byte-5</th>
<th>Byte-6</th>
<th>Byte-7 (MSB)</th>
<th>Lockable</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID</td>
<td>0</td>
<td>UID-0</td>
<td>UID-1</td>
<td>UID-2</td>
<td>UID-3</td>
<td>UID-4</td>
<td>UID-5</td>
<td>UID-6</td>
<td></td>
<td>Locked</td>
</tr>
<tr>
<td>Data</td>
<td>1</td>
<td>Data0</td>
<td>Data1</td>
<td>Data2</td>
<td>Data3</td>
<td>Data4</td>
<td>Data5</td>
<td>Data6</td>
<td>Data7</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>2</td>
<td>Data8</td>
<td>Data9</td>
<td>Data10</td>
<td>Data11</td>
<td>Data12</td>
<td>Data13</td>
<td>Data14</td>
<td>Data15</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>3</td>
<td>Data16</td>
<td>Data17</td>
<td>Data18</td>
<td>Data19</td>
<td>Data20</td>
<td>Data21</td>
<td>Data22</td>
<td>Data23</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>4</td>
<td>Data24</td>
<td>Data25</td>
<td>Data26</td>
<td>Data27</td>
<td>Data28</td>
<td>Data29</td>
<td>Data30</td>
<td>Data31</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>5</td>
<td>Data32</td>
<td>Data33</td>
<td>Data34</td>
<td>Data35</td>
<td>Data36</td>
<td>Data37</td>
<td>Data38</td>
<td>Data39</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>6</td>
<td>Data40</td>
<td>Data41</td>
<td>Data42</td>
<td>Data43</td>
<td>Data44</td>
<td>Data45</td>
<td>Data46</td>
<td>Data47</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>7</td>
<td>Data48</td>
<td>Data49</td>
<td>Data50</td>
<td>Data51</td>
<td>Data52</td>
<td>Data53</td>
<td>Data54</td>
<td>Data55</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>8</td>
<td>Data56</td>
<td>Data57</td>
<td>Data58</td>
<td>Data59</td>
<td>Data60</td>
<td>Data61</td>
<td>Data62</td>
<td>Data63</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>9</td>
<td>Data64</td>
<td>Data65</td>
<td>Data66</td>
<td>Data67</td>
<td>Data68</td>
<td>Data69</td>
<td>Data70</td>
<td>Data71</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>A</td>
<td>Data72</td>
<td>Data73</td>
<td>Data74</td>
<td>Data75</td>
<td>Data76</td>
<td>Data77</td>
<td>Data78</td>
<td>Data79</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>B</td>
<td>Data80</td>
<td>Data81</td>
<td>Data82</td>
<td>Data83</td>
<td>Data84</td>
<td>Data85</td>
<td>Data86</td>
<td>Data87</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>C</td>
<td>Data88</td>
<td>Data89</td>
<td>Data90</td>
<td>Data91</td>
<td>Data92</td>
<td>Data93</td>
<td>Data94</td>
<td>Data95</td>
<td>Yes</td>
</tr>
<tr>
<td>Reserved</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock/Reserved</td>
<td>E</td>
<td>LOOKA</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Reserved for internal use**
- **User Block Lock & Status**
- **OTP bits**
### Appendix E. ACR122 Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>No error.</td>
</tr>
<tr>
<td>01h</td>
<td>Time Out, the target has not answered.</td>
</tr>
<tr>
<td>02h</td>
<td>A CRC error has been detected by the contactless UART.</td>
</tr>
<tr>
<td>03h</td>
<td>A Parity error has been detected by the contactless UART.</td>
</tr>
<tr>
<td>04h</td>
<td>During a MIFARE anti-collision/select operation, an erroneous Bit Count has been detected.</td>
</tr>
<tr>
<td>05h</td>
<td>Framing error during MIFARE operation.</td>
</tr>
<tr>
<td>06h</td>
<td>An abnormal bit-collision has been detected during bit wise anti-collision at 106 kbps.</td>
</tr>
<tr>
<td>07h</td>
<td>Communication buffer size insufficient.</td>
</tr>
<tr>
<td>08h</td>
<td>RF Buffer overflow has been detected by the contactless UART (bit BufferOvfl of the register CL_ERROR).</td>
</tr>
<tr>
<td>0Ah</td>
<td>In active communication mode, the RF field has not been switched on in time by the counterpart (as defined in NFCIP-1 standard).</td>
</tr>
<tr>
<td>0Bh</td>
<td>RF Protocol error (cf. reference [4], description of the CL_ERROR register).</td>
</tr>
<tr>
<td>0Dh</td>
<td>Temperature error: the internal temperature sensor has detected overheating, and therefore has automatically switched off the antenna drivers.</td>
</tr>
<tr>
<td>0Eh</td>
<td>Internal buffer overflow</td>
</tr>
<tr>
<td>10h</td>
<td>Invalid parameter (range, format, etc.)</td>
</tr>
<tr>
<td>12h</td>
<td>DEP Protocol: The chip configured in target mode does not support the command received from the initiator (the command received is not one of the following: ATR_REQ, WUP_REQ, PSL_REQ, DEP_REQ, DSL_REQ, RLS_REQ, ref. [1]).</td>
</tr>
</tbody>
</table>
| 13h        | DEP Protocol/MIFARE/ISO/IEC 14443-4: The data format does not match to the specification. Depending on the RF protocol used, it can be:  
  - Bad length of RF received frame,  
  - Incorrect value of PCB or PFB,  
  - Invalid or unexpected RF received frame,  
  - NAD or DID incoherence. |
<p>| 14h        | MIFARE: Authentication error. |
| 23h        | ISO/IEC 14443-3: UID Check byte is wrong. |
| 25h        | DEP Protocol: Invalid device state, the system is in a state which does not allow the operation. |
| 26h        | Operation not allowed in this configuration (host controller interface). |
| 27h        | This command is not acceptable due to the current context of the chip (Initiator vs. Target, unknown target number, Target not in the good state, etc.). |
| 29h        | The chip configured as target has been released by its initiator. |
| 2Ah        | ISO/IEC 14443-3B only: the ID of the card does not match, meaning that the expected card has been exchanged with another one. |
| 2Bh        | ISO/IEC 14443-3B only: the card previously activated has disappeared. |</p>
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2Ch</td>
<td>Mismatch between the NFCID3 initiator and the NFCID3 target in DEP 212/424 kbps passive.</td>
</tr>
<tr>
<td>2Dh</td>
<td>An over-current event has been detected.</td>
</tr>
<tr>
<td>2Eh</td>
<td>NAD missing in DEP frame.</td>
</tr>
</tbody>
</table>