Table of Contents

1 INTRODUCTION 4
1.1 Purpose 4
1.2 Scope 4
1.3 Glossary 4
1.4 Referenced Document 4

2 PHYSICAL LAYER 5
2.1 Electrical Interface 5
2.2 Data Format 5

3 LINK LAYER 6
3.1 Packet Format 6
3.2 Data Byte Flow Control 7
3.3 Time-out Control 8

4 COMMAND SET 9
4.1 System Commands 13
4.1.1 SetAddress (0x06) 13
4.1.2 SetBaudrate (0x07) 13
4.1.3 GetSerNum (0x09) 14
4.1.4 GetVerNum (0x0A) 14
4.1.5 SetUserInfo (0x0B) 14
4.1.6 GetUserInfo (0x0C) 14

4.2 I/O Commands 15
4.2.1 CMD_Wiegand (0x12) 15
4.2.2 CMD_WiegandActive (0x13) 15
4.2.3 SetLED (0x24) 16
4.2.4 ActiveLED (0x25) 16
4.2.5 ActiveBuzzer (0x26) 16
4.2.6 RF_RST (0x27) 17

4.3 ISO14443 Type-A Commands 18
4.3.1 REQA (0x30) 18
4.3.2 Anticoll of Cascadelevel (0x31) 18
4.3.3 Anticoll of Cascadelevel2 (0x38) 18
4.3.4 Anticoll of Cascadelevel3 (0x3A) 19
4.3.5 Select of Cascadelevel 1 (0x32) 19
4.3.6 Select of Cascadelevel 2 (0x39) 19
4.3.7 Select of Cascadelevel 3 (0x3B) 20
4.3.8 Halt (0x33) 20
4.3.9 CMD_SLE_Gen (0x34) 20
4.4  MIFARE Commands  
4.4.1 MF_Auth (0x40)  
4.4.2 MF_Read (0x41)  
4.4.3 MF_Write (0x42)  
4.4.4 MF_Transfer (0x43)  
4.4.5 MF_Value (0x44)  
4.4.6 MF_LoadKey (0x45)  
4.4.7 MF_LoadEEKey (0x46)  
4.4.8 MF_StoreKeyToEE (0x47)  

4.5  High Level MIFARE Commands  
4.5.1 CMD_MF_HLRead (0x90)  
4.5.2 CMD_MF_HLWrite (0x91)  
4.5.3 CMD_HL_InitVal (0x92)  
4.5.4 CMD_HL_Decrement (0x93)  
4.5.5 CMD_HL_Increment (0x94)  
4.5.6 CMD_HL_Request (0x98)  

5  ERROR/STATUS CODE  

6  REVISION HISTORY  
6.1  Communication Protocol Ver. 2.10  
6.2  Communication Protocol Ver. 2.04  
6.3  Communication Protocol Ver. 2.03  
6.4  Communication Protocol Ver. 2.02  
6.5  Communication Protocol Ver. 2.01  
6.6  Communication Protocol Ver. 0.00
1 Introduction

1.1 Purpose

This document defines a communication protocol, which will be as a generic protocol for products involving data communication with each other. Basically this generic protocol serves for communication between a HOST and one or more terminal devices.

1.2 Scope

Different aspects of the protocol will be described, which include the electrical interface, data format, and link layer. This generic protocol will be applied for

- Point to point – RS232
- Multi-drop (Point to multi-points) – RS422/RS485
- Two wires half-duplex mode and four wires full-duplex mode.

1.3 Glossary

UID – Unique Identification
LRC – Longitudinal Redundancy Check
CRC – Cyclic Redundancy Check
MAC – Message Authentication Code
ATR – Answer To Reset

1.4 Referenced Document

<Not available>
2 Physical Layer

2.1 Electrical Interface

Basically, this communication protocol does not need to be bound with any electrical interface characteristic. Typically the following types of physical link could be used:

- RS232 (Point to point only)
- CMOS-Logic Level (Point to point only)
- Half duplex, two wires RS485/RS422 (multi-drop mode supported)
- Full duplex, four wires RS485/RS422 (multi-drop mode supported)

2.2 Data Format

The data format (Start Bit, Data Bits, parity, Stop Bit) is software configurable, and can be set to match the special requirement of data transmission between two communication devices. The general data format is defined as:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud Rate</td>
<td>Selective: 9600, 19200, 38400, 57600, 115200</td>
</tr>
<tr>
<td>Data Bits</td>
<td>Fixed: 8 bits</td>
</tr>
<tr>
<td>Start Bit</td>
<td>Fixed: 1 Bits</td>
</tr>
<tr>
<td>Stop Bit</td>
<td>Selective: 1 bit.</td>
</tr>
<tr>
<td>Parity</td>
<td>Selective: Odd, Even, None</td>
</tr>
</tbody>
</table>

The following is the default setting:

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Data Bits</th>
<th>Start Bit</th>
<th>Stop Bit</th>
<th>Parity</th>
</tr>
</thead>
<tbody>
<tr>
<td>115200</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>None</td>
</tr>
</tbody>
</table>

The configuration for the communication setting will be stored within the reader’s on board EEPROM memory and can be changed by command sent from the Host.
3 Link Layer

The communication protocol is a packet-oriented protocol - all the data exchanged between two communication devices will be based on packet format. The protocol is designed for multi-drop mode and where point-to-point mode could be treated as a special case of multi-drop mode.

The data packet starts with the control character ‘STX’ and ends with ‘ETX’, which follows the 8-bit BCC checksum. Besides the checksum is used for error checking, character (byte) time-out and packet (command) time-out are used to re-synchronous the communication.

3.1 Packet Format

There are two types of data packets. Command Message is the packet sent from the Host to the reader device. The Reply Message is the packet sent from the reader to the Host.

Packet format for Command Message (Host to Reader)

(BCC) = SEQ ⊕ DADD ⊕ CMD ⊕ DATA LENGTH ⊕ TIME ⊕ DATA[0] ⊕ … ⊕ DATA[n], where ⊕ is the “EOR”.

Packet format for Reply Message (Reader to Host)

(BCC) = SEQ ⊕ DADD ⊕ DATA LENGTH ⊕ STATUS ⊕ DATA[0] ⊕ … ⊕ DATA[n], where ⊕ is the “EOR”.

The following table describes the packet fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>STX</td>
<td>1</td>
<td>0x02 - &quot;Start of Text’ – standard control character. It is the starting of a data packet.</td>
<td></td>
</tr>
<tr>
<td>SEQ</td>
<td>1</td>
<td>Packet sequence number: this field acts as an error control. Each packet sent from the Host associates with a sequence number that will be increased circularly. The reader returns the reply message with the same SEQ number. The HOST can check the SEQ for the occurrence of the ‘OUT of SEQUENCE’ error.</td>
<td>The address extend bits are used to widen the device address range in case 255 addresses are not enough.²</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: Always set to ‘1’</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6-4: Sequence Number. Change from 0 to 7 cyclically.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 3-0: Extend Device Address.</td>
<td></td>
</tr>
<tr>
<td>DADD</td>
<td>1</td>
<td>Device Address, which is used for multi-drop mode, only the reader (device) with matched pre-</td>
<td>Address 0x00 is a special address for</td>
</tr>
</tbody>
</table>

¹ The SEQ Number is not handled at this version.

² The Address Extend Bits are reserved for future use. At this moment, the extend bits have no effect.
programmed device address will response the received command packet.

point-to-point mode communication. The reader responds to all the packets which has a “0” address. (No Address matching checking will be made.\(^1\)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CMD</strong></td>
<td>1</td>
<td>Command field: the command field consists of one command byte.</td>
</tr>
<tr>
<td><strong>DATA LENGTH</strong></td>
<td>1</td>
<td>Length of the data bytes in the packet. The Data Length includes the TIME/STATUS and the DATA field, but not the BCC. LENGTH= Number_of_Bytes (TIME/STATUS + DATA[0..N] )</td>
</tr>
<tr>
<td><strong>STATUS</strong></td>
<td>1</td>
<td>Reply Status byte: The status replied from Reader to Host</td>
</tr>
<tr>
<td><strong>TIME</strong></td>
<td>1</td>
<td>Extend the time-out delay for special commands that may need extra time to process. For most of the commands, this byte is set to 0.</td>
</tr>
<tr>
<td><strong>DATA [0-N]</strong></td>
<td>0 – 80</td>
<td>The Data Field is a stream of data with variable length, which depends on the Command word. There are also some COMMANDs have zero length of data field. If the Data Field of the Command/Reply Message has more then 80 bytes, the reader won’t response and treats this command as an error and wait for another command.</td>
</tr>
<tr>
<td><strong>BCC</strong></td>
<td>1</td>
<td>Eight-bit block check sum. The calculation of the check sum includes all the bytes within the package but excludes the STX, ETX.</td>
</tr>
<tr>
<td><strong>ETX</strong></td>
<td>1</td>
<td>0x03:’END of TEXT’ – standard Control Character which indicates the END of a packet. Before the ETX, an 8-bit BCC Check Sum byte is inserted for data integration checking.</td>
</tr>
</tbody>
</table>

### 3.2 Data Byte Flow Control

At this moment, no flow control will be considered.

---

\(^1\) A command packet with address “0” may cause communication problem if more then one readers are connected as multi-drop mode.
A time-out mechanism is used for the packet flow control. There are two levels time-out control: byte time-out and packet time-out.

- The byte time-out is the maximum time delay between transmitted bytes. The interval may be varied from a few ms to tens of ms. (Default 30ms)

- Packet (command) Time-out: the maximum period allows for transmitting a complete packet. This time-out interval may be within the range of tens of millisecond to hundreds of millisecond. It depends on the real application and the possible largest packet size. The duration of the packet time-out is implied by the command itself. (Some commands may have shorter time-out then other commands)
4 Command Set

The commands are grouped to different categories. They are System command, I/O commands, ISO14443 standard commands, MIFARE commands, SAM commands and miscellaneous commands.

<table>
<thead>
<tr>
<th>CMD Code</th>
<th>Name</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x06</td>
<td>SetAddress</td>
<td>Program the Device Address to the reader</td>
<td></td>
</tr>
<tr>
<td>0x07</td>
<td>SetBaudrate</td>
<td>Set the reader’s communication baud rate</td>
<td></td>
</tr>
<tr>
<td>0x09</td>
<td>GetSerlNum</td>
<td>Get the reader’s Serial Number</td>
<td></td>
</tr>
<tr>
<td>0x0A</td>
<td>GetVerNum</td>
<td>Get the reader’s firmware version number.</td>
<td></td>
</tr>
<tr>
<td>0x0B</td>
<td>SetUserInfo</td>
<td>Set the Usr Information</td>
<td>A 32-byte memory space is reserved for storing the programmable user information, which could be used for customer service records or tacking of project.</td>
</tr>
<tr>
<td>0x0C</td>
<td>GetUserInfo</td>
<td>Get the User Information</td>
<td></td>
</tr>
<tr>
<td>0x11,0x08</td>
<td>Internal Commands</td>
<td>Internal Commands, not open for end users</td>
<td>These commands are for internal testing use only. Misuse of these commands may damage the reader.</td>
</tr>
<tr>
<td>0x12</td>
<td>CMD_Wiegand</td>
<td>Enter Wiegand status, and save the setting</td>
<td></td>
</tr>
<tr>
<td>0x13</td>
<td>CMD_WiegandActive</td>
<td>Wiegand status</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CMD Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x21</td>
<td>SetPort</td>
<td>Set the CV3500A I/O ports</td>
</tr>
<tr>
<td>0x22</td>
<td>GetPort</td>
<td>Read status of the I/O ports.</td>
</tr>
<tr>
<td>0x24</td>
<td>SetLED</td>
<td>Turn On/Off the LEDs</td>
</tr>
</tbody>
</table>

1 Only CMD0 used, CMD1 always set to 0.
### ISO14443 TTYPE A Commands (0x30-0x3F)

<table>
<thead>
<tr>
<th>CMD Code</th>
<th>Name</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30</td>
<td>REQA</td>
<td>ISO14443A REQUEST</td>
<td></td>
</tr>
<tr>
<td>0x31</td>
<td>Anticoll of Cascadelevel1</td>
<td>ISO14443A Anti-collision</td>
<td></td>
</tr>
<tr>
<td>0x32</td>
<td>Select of Cascadelevel1</td>
<td>ISO14443A Select</td>
<td></td>
</tr>
<tr>
<td>0x33</td>
<td>Halt</td>
<td>ISO14443A Halt</td>
<td></td>
</tr>
<tr>
<td>0x34</td>
<td>SLE_GEN</td>
<td>Send generic (transparent)</td>
<td>Internal Command, not open to end-user yet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>command to the SLE55Rxx card</td>
<td></td>
</tr>
<tr>
<td>0x35</td>
<td>MF_PowerSave</td>
<td>Let the Module Enter the</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Power Saving mode.</td>
<td></td>
</tr>
<tr>
<td>0x38</td>
<td>Anticoll of Cascadelevel2</td>
<td>ISO14443A Anti-collision</td>
<td></td>
</tr>
<tr>
<td>0x39</td>
<td>Select of Cascadelevel2</td>
<td>ISO14443A Select</td>
<td></td>
</tr>
<tr>
<td>0x3a</td>
<td>Anticoll of Cascadelevel3</td>
<td>ISO14443A Anti-collision</td>
<td></td>
</tr>
<tr>
<td>0x3b</td>
<td>Select of Cascadelevel3</td>
<td>ISO14443A Select</td>
<td></td>
</tr>
</tbody>
</table>

### Mifare Commands (0x40-0x4F)

<table>
<thead>
<tr>
<th>CMD Code</th>
<th>Name</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x40</td>
<td>MF_Auth</td>
<td>Mifare Authentication</td>
<td>The key must be preloaded to the RC500’s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Master Key buffer</td>
</tr>
<tr>
<td>0x41</td>
<td>MF_Read</td>
<td>Mifare Read</td>
<td></td>
</tr>
<tr>
<td>0x42</td>
<td>MF_Write</td>
<td>MIFARE Write</td>
<td></td>
</tr>
<tr>
<td>0x43</td>
<td>MF_Transfer</td>
<td>MIFARE Value Block Transfer</td>
<td></td>
</tr>
<tr>
<td>0x44</td>
<td>MF_Value</td>
<td>MIFARE Value related command</td>
<td>Three sub-commands provided: increase,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>decrease, restore.</td>
</tr>
<tr>
<td>0x45</td>
<td>MF_Loadkey</td>
<td>Load a key string to the Master</td>
<td></td>
</tr>
<tr>
<td>CMD Code</td>
<td>Name</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>0x46</td>
<td>MF_LoadEEKey</td>
<td>Load a key from RC500’s EEPROM to Master Key buffer</td>
<td></td>
</tr>
<tr>
<td>0x47</td>
<td>MF_StoreKeyToEE</td>
<td>Store a key to MFRC500 chip’s internal EEPROM.</td>
<td></td>
</tr>
</tbody>
</table>

### ISO14443-B Command (0x60-0x6F)

<table>
<thead>
<tr>
<th>CMD Code</th>
<th>Name</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x60</td>
<td>Request_B</td>
<td>ISO14443B REQB command</td>
<td></td>
</tr>
<tr>
<td>0x61</td>
<td>Anticoll_B</td>
<td>Start the Anticollision loop for the Type B Card</td>
<td></td>
</tr>
<tr>
<td>0x62</td>
<td>Attrib_B</td>
<td>ISO14443B ATTRIB command</td>
<td></td>
</tr>
<tr>
<td>0x63</td>
<td>Halt_B</td>
<td>ISO14443B Halt command</td>
<td></td>
</tr>
<tr>
<td>0x65</td>
<td>Generic_B</td>
<td>ISO14443-4 transparent command Type B Card</td>
<td></td>
</tr>
<tr>
<td>0x64</td>
<td>FoundCards</td>
<td>Automatically detect which type (A/B) of card is in the field.</td>
<td></td>
</tr>
</tbody>
</table>

### SAM Commands (0x70 –0x7F)

<table>
<thead>
<tr>
<th>CMD Code</th>
<th>Name</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x70</td>
<td>SAM_ATR</td>
<td>Get Answer-to-reset String from the SAM card</td>
<td>Not supported yet</td>
</tr>
<tr>
<td>0x71</td>
<td>SetBaudRateSAM</td>
<td>Set the communication baud rate for SAM module interface</td>
<td>Not supported yet</td>
</tr>
<tr>
<td>0x72</td>
<td>SAM_GEN</td>
<td>Send generic (transparent) command to the SAM card.</td>
<td>Use ISO7816 T=1 protocol</td>
</tr>
</tbody>
</table>

### Miscellaneous Commands (0xC0-0xCF)

<table>
<thead>
<tr>
<th>CMD Code</th>
<th>Name</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x90</td>
<td>MF_HLRead</td>
<td>The High Level Read command integrates the low level commands (request, anti-collision, select, authentication, read) to achieve the reading operation with a one-step single command.</td>
<td>High level command</td>
</tr>
<tr>
<td>0x91</td>
<td>MF_HLWrite</td>
<td>The High Level Write command integrates the low level commands (request, anti-collision, select, authentication, write) to achieve the writing operation with a one-step single command.</td>
<td>High level command</td>
</tr>
<tr>
<td>0x92</td>
<td>MF_HLInitVal</td>
<td>The High Level Value Block Initialization command integrates the low level commands (request, anti-collision, select, authentication) to achieve the</td>
<td>High level command</td>
</tr>
</tbody>
</table>
value block initialization with a one-step single command.

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x93</td>
<td>MF_HLDecrement</td>
<td>The High Level Decrement command integrates the low level commands (request, anti-collision, select, authentication) to achieve the Decrement with a one-step single command.</td>
</tr>
<tr>
<td>0x94</td>
<td>MF_HLIncrement</td>
<td>The High Level Increment command integrates the low level commands (request, anti-collision, select, authentication) to achieve the Increment with a one-step single command.</td>
</tr>
<tr>
<td>0x98</td>
<td>MF_HLRequest</td>
<td>The High Level Request command integrates the low level commands (request, anticoll, select, anticoll2, select2, anticoll3, select3) to achieve the select card with a one-step single command.</td>
</tr>
</tbody>
</table>

**Debug Commands - For Internal use only (0xD0-0xDF)**

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xD0</td>
<td>ReadEEPROM</td>
<td>Read data from the RC500’s internal EEPROM.</td>
<td>Only using for interior debug, no open to customer.</td>
</tr>
<tr>
<td>0xD1</td>
<td>WriteEEPROM</td>
<td>Write data to the RC500’s internal EEPROM.</td>
<td>Only using for interior debug, no open to customer.</td>
</tr>
<tr>
<td>0xD2</td>
<td>EraseEEPROM</td>
<td>Erasing the RC500’s internal EEPROM.</td>
<td>Only using for interior debug, no open to customer.</td>
</tr>
</tbody>
</table>

1 The debugging commands will subject to change without notice.
4.1 System Commands

4.1.1 SetAddress (0x06)

Data Field

DATA[0]: Enable Serial Number Checking
- 0x00 – Don’t check the Serial Number.
- 0x01 – Check the Serial Number.

DATA[1]: The new Device (Reader) Address to be set

DATA[2..9]: Reader Serial Number.

Response:

STATUS: 0x00 – OK

Data Field

DATA[0]: The programmed device address.

Description

Program a device address to the reader and returns new device address. In order to program the device address successfully, the correct reader’s pre-programmed Serial Number must be submitted. (unless the Serial Number Checking Flag is disabled).

4.1.2 SetBaudrate (0x07)

Data Field

DATA[0]: Communication speed
- 0x00 – 9600 bps
- 0x01 – 19200 bps
- 0x02 – 38400 bps
- 0x03 – 57600 bps
- 0x04 – 76800 bps
- 0x05 – 115200 bps

Response:

STATUS: 0x00 - OK

Data Field

DATA[0]: Return the new communication speed.
- 0x00 – 9600 bps
- 0x01 – 19200 bps
- 0x02 – 38400 bps
- 0x03 – 57600 bps
- 0x04 – 76800 bps
- 0x05 – 115200 bps

Description

Set the reader’s baud rate for host communication. The baud rate will be stored in the reader’s EEPROM and used as the new default baud rate.

Note: The new baud rate will not take effect until the reader is reset.
4.1.3 GetSerNum (0x09)

Data Field: N/A

Response:

STATUS: 0x00 – OK

Data Field:

DATA[0..7]: 8-byte Reader Serial Number
DATA[8..9]: Device Address

Description:
Get the Device Address and the Serial Number from the reader. The 8-byte serial number is pre-programmed by the factory.

4.1.4 GetVerNum (0x0A)

Data Field: N/A

Response:

STATUS: 0x00 – OK

Data Field:

DATA[0]: Device Address
DATA[1..N]: The Version Number of the Reader’s Firmware

Description:
Get the reader’s firmware version number.

4.1.5 SetUserInfo (0x0B)

Data Field:

DATA[0..31]: 32 bytes user information.

Response:

STATUS: 0x00 – OK

Data Field: n.a.

Description:
Program the 32 bytes information to the reader.

4.1.6 GetUserInfo (0x0C)

Data Field: N/A

Response:

STATUS: 0x00 – OK

Data Field
DATA[0..31]  32 bytes of user information returned

Description
Get the 32-byte user information from the reader.

4.2 I/O Commands

4.2.1 CMD_Wiegand (0x12)

Data Field:

DATA[0]  Wiegand status

bit0  1 ----- Buzzer and LED controlled by external I/O
       0 ----- Buzzer and LED not 1 controlled by external I/O

bit1  1 ----- Prompt from Buzzer and LED after successful read
       0 ----- No prompt from Buzzer and LED after successful read

bit4  1 ----- Enable Wiegand
       0 ----- Disable Wiegand

Response
STATUS:  0x00 – OK

Description
Set Wiegand Status of the reader, the reader will operate accordingly, and the setting will be stored in the reader.

4.2.2 CMD_WiegandActive (0x13)

DATA[0]  Wiegand status

bit0  1 ----- Buzzer and LED controlled by external I/O controlled
       0 ----- Buzzer and LED not 1 controlled by external I/O

bit1  1 ----- Prompt from Buzzer and LED after successful read
       0 ----- No prompt from Buzzer and LED after successful read

bit4  1 ----- Enable Wiegand
       0 ----- Disable Wiegand

Response
STATUS:  0x00 – OK

Description
Set Wiegand Status of the reader, the reader will operate accordingly
4.2.3 SetLED (0x24)

Data Field

DATA[0]: The two LEDs are turned on/off according to the corresponding bits

Bit 0 - Red LED (1=on, 0= off)
Bit 1 - Green LED (1= on, 0= off)

Response:

STATUS: 0x00 - OK
Data Field: N/A

Description:

Turn on/off the LEDs.

4.2.4 ActiveLED (0x25)

Data Field

DATA[0]: Select LED
0x01 - Red LED
0x02 - Green LED
0x03 – Both Red & Green LED.

DATA[1]: Units of on time. Each unit is 100ms.

DATA[2]: Number of cycles to turn on/of the LED. The cycle time is one second.

0xFF will toggle the LED continually.

Response:

STATUS: 0x00 - OK
Data Field: N/A

Description:

The selected LED will be toggled (turn on and off) with the cycles selected and the on time is defined by DATA[1]. For example, if DATA[1] = 4 and DATA[2] = 3, the selected LED will be toggled three times and each time the LED will be turned on for 400ms.

4.2.5 ActiveBuzzer(0x26)

Data Field

DATA[0]: Mode Control
0 – Turn off the buzzer
1 – Turn on the buzzer.
4 – Play a sound pattern. The sound pattern is a sequence of on-off-on-off sound and controlled by DATA[1..5].

DATA[1]: Units of first on time. Each unit is 100ms.

DATA[2]: Units of first off time. Each unit is 100ms.

DATA[3]: Units of second on time. Each unit is 100ms.

DATA[4]: Units of second off time. Each unit is 100ms.
DATA[5]: Cycle times.

Response:
STATUS: 0x00 - OK
Data Field: N/A

Description:
Control the buzzer to play sound patterns.

4.2.6 **RF_RST (0x27)**

Data Field:
DATA[0]: The units of time to turn off the RF field. Each time unit is equal 100us.

Response:
STATUS: 0 – OK
DATA Field: N/A

Description:
The RF field will be turn off (reset) and then the field will be turned on again. The value 0x00 will turn off the field forever until another RF_RST is issued.
4.3 ISO14443 Type-A Commands

4.3.1 REQA (0x30)

Data Field

DATA[0]: Request mode
0x26 – Request Idle
0x52 – Request All (Wake up all)

Response:
STATUS: 0x00 - OK
DATA[0..1]: The two-bytes ATQ response from the card.

Description
Send the ISO14443 A REQUEST command to the card.

4.3.2 Anticoll of Cascadelevel (0x31)

Data Field: N/A

Response:
STATUS: 0x00 - OK

Data Field

DATA[0..3]: UID – the card serial number
DATA[4]: Multi-card flag.
0x00 - One cared detected.
0x01 - Multiple cards detected.

Description:
Execute the ISO14443 Type A Anti-collision loop of cascadelevel1. The card’s UID (serial number) of cascadelevel1 will be returned. If more then one cards are detected in the field, the Multi-Card Flag will be set.

Note: only cards not halted will be detected by the Anti-collision loop.

4.3.3 Anticoll of Cascadelevel2 (0x38)

Data Field: N/A

Response:
STATUS: 0x00 - OK

Data Field

DATA[0..3]: UID – the card serial number
DATA[4]: Multi-card flag.
0x00 - One cared detected.
0x01 - Multiple cards detected.

Description:
Execute the ISO14443 Type A Anti-collision loop of cascadelevel2. The card’s UID (serial number) of cascadelevel2 will be returned. If more than one cards are detected in the field, the Multi-Card Flag will be set.

Note: only cards not halted will be detected by the Anti-collision loop.

4.3.4 Anticoll of Cascadelevel3 (0x3a)

Data Field: N/A

Response:

STATUS: 0x00 - OK

Data Field

DATA[0..3]: UID – the card serial number

DATA[4]: Multi-card flag.

0x00 - One card detected.
0x01 - Multiple cards detected.

Description:

Execute the ISO14443 Type A Anti-collision loop of cascadelevel3. The card’s UID (serial number) of cascadelevel3 will be returned. If more than one cards are detected in the field, the Multi-Card Flag will be set.

Note: only cards not halted will be detected by the Anti-collision loop.

4.3.5 Select of Cascadelevel 1 (0x32)

Data Field

DATA[0..3]: UID – the UID of the card to be selected.

Response:

STATUS: 0x00 - OK

0x46 - OK, but need next anticoll-select loop

Data Field

DATA[0..3]: UID – the UID of the card to be selected.

Description:

ISO14443 A SELECT of Cascadelevel1 command.

4.3.6 Select of Cascadelevel 2 (0x39)

Data Field

DATA[0..3]: UID – the UID of the card to be selected.

Response:

STATUS: 0x00 - OK

0x46 - OK, but need next anticoll-select loop

Data Field
DATA[0..3]: UID – the UID of the card to be selected.

Description:
ISO14443 A SELECT of Cascadelevel2 command.

4.3.7 Select of Cascadelevel 3 (0x3b)

Data Field
DATA[0..3]: UID – the UID of the card to be selected.

Response:
STATUS: 0x00 - OK
Data Field
DATA[0..3]: UID – the UID of the card to be selected.

Description:
ISO14443 A SELECT of Cascadelevel3 command.

4.3.8 Halt (0x33)

Data Field

Response:
STATUS: 0x00
Data Field: N/A

Description:
ISO14443 A Halt command

4.3.9 CMD_SLE_Gen (0x34)

CTRL: CRC Enable Flag.
0x00 – No CRC checksum will be calculated and appended.
0x01 – The CRC checksum will be calculated and appended.

Data Field
DATA[0]: Data length to be transmitted.
DATA[1..N] Data to be transmit.

Response:
STATUS: 0x00 - OK
Data Field
DATA[0] The length of the data returned from the SLE55Rxx card.
DATA[1..N] Data returned from the SLE55Rxx card.

Description:
Send a transparent command to the SLE55Rxx. The 16-bit CRC checksum will be appended automatically if the CRC Enable flag is set.

4.4 MIFARE Commands

4.4.1 MF_Auth (0x40)

Data Field

DATA[0]: Authentication Mode – Select Key A mode or Key B mode
0x60 – the KEY will be used as KEYA for authentication
0x61 – the KEY will be used as KEYB for authentication.

DATA[1..4]: UID (Card Serial Number)

DATA[5]: The Address of the memory block to be authenticated

Response:

STATUS: 0x00 - OK

Data Field: N/A

Description:

Execute a mutual authentication between the MIFARE card and the reader. The read/write/value commands only can be executed after the successful authentication. The corresponding authentication key should be loaded to the Master Key Buffer by the LOADKEY command. The Authentication Mode determines the key loaded acts as KEY A or KEY B.

Note: before execute the AUTH command, the corresponding key must be loaded to the Master Key Buffer. (either by the LoadKey or LoadEEKey command)

4.4.2 MF_Read (0x41)

Data Field

DATA[0]: Starting Block address

DATA[1]: number of blocks to be read (max. 4 blocks.).

Response:

STATUS: 0x00 - OK

Data Field

DATA[0 .. N]: Data read from the MIFARE card.

Description:

Read multiple memory blocks from the MIFARE Card. Up to four blocks could be read by a single command.

4.4.3 MF_Write (0x42)

Data Field

DATA[0]: Starting Block address

DATA[1]: number of blocks to be written (max. 4 blocks.)

DATA[2..N]: Data to be written
Response:
STATUS: 0x00 - OK
Data Field N/A

Description:
Write multiple memory blocks to the MIFARE Card. The number of blocks could be up to four blocks.

4.4.4 **MF_Transfer (0x43)**

Data Field
DATA[0]: Address of the value block

Response:
STATUS: 0x00 - OK
Data Field

Description:
Transfer command for the MIFARE card. Transfer the value (updated by the increment/decrement command) from the MIFARE chip’s internal value buffer to the real value block.

4.4.5 **MF_Value (0x44)**

Data Field
DATA[0]: Mode – function select - increment/decrement/restore
  0xC0: Decrement
  0xC1: Increment
  0xC2: Restore
  0xC3: ReadValue
    - Read back the value (not in value block format) from the value block
  0xC4: InitValue
    - Initialize a value block.
DATA[1]: Address of the value block
DATA[2..5]: Value – the value to be increased, decreased or initialize. A zero value should be used for RESTORE.

Response:
STATUS: 0x00 - OK
Data Field
DATA[0-3]: Return value, only valid for the ReadValue sub-command, the returned value will be always zero for other sub-commands.

Description:
MIFARE’s value block related functions. There are five sub-functions (restore, increment, decrement, initialize value, read value) under the VALUE command.

4.4.6 **MF_LoadKey (0x45)**

Data Field
DATA[0..5]: The un-coded 6-bytes key string. (i.e. A0A1A2A3A4A5)
4.4.7 MF_LoadEEKey (0x46)

Data Field

DATA[0]: KEYA_B : Select to load KeyA or KeyB
0x60: Load KeyA to the Master Key Buffer
0x61: Load KeyB to the Master Key Buffer

DATA[1]: Sector/Key Number [00..15]. (The number/sector of the key to be loaded)

Response:

STATUS: 0x00 – OK

Data Field

Description:

Load the Key string directly to the Master Key buffer of the MFRC500 chip.

4.4.8 MF_StoreKeyToEE (0x47)

Data Field

DATA[0]: KEYA_B : Select to load KeyA or KeyB
0x60: Load KeyA to the Master Key Buffer
0x61: Load KeyB to the Master Key Buffer

DATA[1]: Sector/Key Number [00..15]. (The sector/key number key to be stored)

DATA[2..7]: un-coded 6-byte key string (i.e. A0A1A2A3A4A5 )

Response:

STATUS: 0x00 – OK

Data Field

Description:

Store the Key to MFRC500’s internal EEPROM.
4.5 High Level MIFARE Commands

4.5.1 CMD_MF_HLRead (0x90)

Data Field

DATA[0]: Mode Control

  Bit0 : Request Mode. 0 = Request Idle, 1 = Request All
  Bit1 : Compare Card Serial Number.
          If this bit is enable, the detected card serial number will be compared with the submitted card serial number (DATA[3-6]). The high level command will be continued only if the serial number is matched.
          0 = Enable, 1 = Disable.

  Bit2 : Key Select. Select use KeyA or Key B for Authentication
          0 = KeyA, 1 = KeyB

  Bit3-7 : Reserved for future used.

DATA[1]: Number of blocks to be read (Max 4)

DATA[2]: The Start Address of blocks to be read.

DATA[3-6]: Card Serial Number (LL LH HL HH)

  The Serial Number will be ignored if Bit 1 of Mode Control (DATA[0]) is not set.

Response:

Data Field

STATUS: 0x00 – OK

DATA[0-3]: Card Serial Number (LL LH HL HH)

DATA[4..N]: Data read from the card.

Description:

The High Level Read Command integrates the low level commands (Request, Anti-Collision, Select, Authentication, ..) and let the user to open the card and read data from the memory blocks by a single command. The high level command select the key stored in the RC500’s internal EEPROM according to the block address. (i.e- Sector Number) and the Mode Control parameter determine the behavior of the high level command.

4.5.2 CMD_MF_HLWrite (0x91)

Data Field

DATA[0]: Mode Control – Refer the CMD_MFRead Command

DATA[1]: Number of blocks to be written (Max 4)

DATA[2]: The Start Address of blocks to be written.

DATA[3-6]: Card Serial Number (LL LH HL HH)

  The Serial Number will be ignored if Bit 1 of Mode Control (DATA[0]) is not set.

DATA[7-N]: Data to be written to the memory blocks.
The High Level Write Command integrates the low level commands (Request, Anti-Collision, Select, Authentication, ..) and let the user to open the card and write data to the memory blocks by a single command. The high level command select the key stored in the RC500’s internal EEPROM according to the block address. (i.e- Sector Number) and the Mode Control parameter determine the behavior of the high level command.

4.5.3 **CMD_HL_InitVal (0x92)**

Data Field

DATA[0]: Mode Control – Refer the CMD_MFRead Command

DATA[1]: Reserved for future used. Set to 0 as default value

DATA[2]: The Sector used for Value storage.

- Block0 –Opened for user use.
- Block1 –Value Stored Block
- Block2 –Value Backup Block.

DATA[3-6]: Card Serial Number (LL LH HL HH)

The Serial Number will be ignored if Bit 1 of Mode Control (DATA[0]) is not set.

DATA[7-10]: The initial value to be stored to the value block. (Value format : LL LH HL HH)

Response:

Data Field

STATUS: 0x00 – OK

DATA[0-3]: Card Serial Number ( LL LH HL HH )

Description:

The High Level Value Initialization Command integrates the low level commands (Request, Anti-Collision, Select, Authentication, ..) and let the user to initialize a sector for value storage use. The high level command select the key stored in the RC500’s internal EEPROM according to the Sector number. The Mode Control parameter determines the behavior of the high level command.

4.5.4 **CMD_HL_Decrement (0x93)**

Data Field

DATA[0]: Mode Control – Refer the CMD_MFRead Command

DATA[1]: Reserved for future used. Set to 0 as default value

DATA[2]: The Sector Number of the Value Sector.

DATA[3-6]: Card Serial Number (LL LH HL HH)
The Serial Number will be ignored if Bit 1 of Mode Control (DATA[0]) is not set.

**DATA[7-10]:** The value to be decreased to the value block. (Value format: LL LH HL HH)

**Response:**

Data Field

**STATUS:** 0x00 – OK

**DATA[0-3]:** Card Serial Number (LL LH HL HH)

**DATA[4-7]:** Value after decreased (LL LH HL HH)

**Description:**

The High Level Value Decrement Command integrates the low level commands (Request, Anti-Collision, Select, Authentication, ..) and let the user to decrease the selected value.

Note: The selected sector should be pre-initialized by the High Level Value Initialization command.

---

**4.5.5 CMD_HL_Increment (0x94)**

Data Field

**DATA[0]:** Mode Control – Refer the CMD_MFRead Command

**DATA[1]:** Reserved for future used. Set to 0 as default value

**DATA[2]:** Sector Number of the value sector.

**DATA[3-6]:** Card Serial Number (LL LH HL HH)

The Serial Number will be ignored if Bit 1 of Mode Control (DATA[0]) is not set.

**DATA[7-10]:** The value to be increased to the value block. (Value format: LL LH HL HH)

**Response:**

Data Field

**STATUS:** 0x00 – OK

**DATA[0-3]:** Card Serial Number (LL LH HL HH)

**DATA[4-7]:** Value after increased (LL LH HL HH)

**Description:**

The High Level Value Increment Command integrates the low level commands (Request, Anti-Collision, Select, Authentication,) and let the user to increase the selected value sector.

Note: The selected sector should be pre-initialized by the High Level Value Initialization command.

---

**4.5.6 CMD_HL_Request (0x98)**

Data Field

**DATA[0]:** Request mode
0x00 – Request Idle

0x01 – Request All (Wake up all)

**Response:**

Data Field

**STATUS:** 0x00 – OK

**DATA[0-10]:** Card Serial Number

**Description:**

The High Level Value Increment Command integrates the low level commands (Request, AntiColl1, Select1, AntiColl2, Select2, AntiColl3, Select3) and get the SNR of selected card (4bytes for selectlevel1, 7bytes for selectlevel2, 10bytes for selectlevel3).
5 Error/Status Code

System Error/Status Codes (0x00-0x0F)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>OK, Command OK. (success)</td>
</tr>
<tr>
<td>0x01</td>
<td>PARA_ERR, Parameter value out of range error</td>
</tr>
<tr>
<td>0x04</td>
<td>TMO_ERR, Reader reply time out error</td>
</tr>
<tr>
<td>0x05</td>
<td>SEQ_ERR, Communication Sequence Number out of order</td>
</tr>
<tr>
<td>0x06</td>
<td>CMD_ERR, Reader received unknown command</td>
</tr>
<tr>
<td>0x07</td>
<td>CHKSUM_ERR, Communication Check Sum Error</td>
</tr>
<tr>
<td>0x08</td>
<td>INTR_ERR, Unknown Internal Error</td>
</tr>
</tbody>
</table>

Card Error/Status Codes (0x10-0x1F)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x11</td>
<td>NOTAG_ERR, No card detected</td>
</tr>
<tr>
<td>0x12</td>
<td>CRC_ERR, Wrong CRC received from card</td>
</tr>
<tr>
<td>0x13</td>
<td>PARITY_ERR, Wrong Parity Received from card</td>
</tr>
<tr>
<td>0x14</td>
<td>BITCNT_ERR, Wrong number of bits received from the card</td>
</tr>
<tr>
<td>0x15</td>
<td>BYTECNT_ERR, Wrong number of bytes received from the card</td>
</tr>
<tr>
<td>0x16</td>
<td>CRD_ERR, Any other error happened when communicate with card</td>
</tr>
</tbody>
</table>

MIFARE Error/Status Codes (0x20-0x2F)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>MF_AUTHERR, No Authentication Possible</td>
</tr>
<tr>
<td>0x21</td>
<td>MF_SERNRERR, Wrong Serial Number read during Anti-collision.</td>
</tr>
<tr>
<td>0x22</td>
<td>MF_NOAUTHERR, Card is not authenticated</td>
</tr>
<tr>
<td>0x23</td>
<td>MF_VALFMT, Not value block format</td>
</tr>
<tr>
<td>0x24</td>
<td>MF_VAL, Any problem with the VALUE related function</td>
</tr>
</tbody>
</table>

Type-B Card Error/Status Codes (0x30-0x3F)

<To be defined>

SAM Error/Status Codes (0x40-0x4F)

<To be defined>
6 Revision History

6.1 Communication Protocol Ver. 2.10
Date 08/31/2004
- Add I/O Commands, e.g. Wiegand, WiegandActive

6.2 Communication Protocol Ver. 2.04
Date 05/07/2003
- Add AnticollLevel2,3 & SelectLevel2,3 & HighLevel Request command

6.3 Communication Protocol Ver. 2.03
Date 04/25/2003
- Correct the mistakes in the following commands:
  - CMD SetAddress (0x06) - swap Data[0] and Data[1]
  - CMD SetBaudRate (0x07) – the new baud rate will be returned in the reply message
  - CMD RF_RST (0x27) – the reset time of the RF field is fixed.
  - CMD Select (0x32) - the card number will be returned in the reply message
  - High Level Mifare Commands: the parameter “MODE”
    - Bit 0 - “REQUEST ALL” Flag
    - Bit 2 – “KeyA/B” Flag

6.4 Communication Protocol Ver. 2.02
Date 03/02/2003
- Error correction
- Add Cover Sheet and Table of Contents

6.5 Communication Protocol Ver. 2.01
Date 24/02/2003
- Add the High Level Commands

6.6 Communication Protocol Ver. 0.00
Date 26/09/2002 Ver0.01 (Internal Draft)
Date 23/09/2002 Ver0.00 (First Draft)
- Reference: Communication Protocol for the SR170 Reader (UM-Comm-ProtocolV2.02)