PMA13292-AA
DRAFT - PUBLIC

SpringCard FunkyGate-DW NFC

Integration and Configuration Guide

www.springcard.com

Document Identification

Category	Owner's Manual		
Family/Customer	FunkyGate NFC Series		
Reference	PMA13292	Version	AA
Status	draft	Classification	Public
Keywords			
Abstract			

File name	[PMA13292-AA] FunkyGate-DW Integration and Configuration Guide.odt		
Date saved	17/01/14	Date printed	18/12/13

Revision History

Ver.	Date	Author	Valid. by		Approv. by	Details
			Tech.	Qual.		
AA	17/01/14	JDA				First release, created from PMA8P3P

Contents

1. Introduction

1.1. Abstract

SpringCard FunkyGate-DW NFC is a RFID (13.56MHz) and NFC wall-mount Reader, for access control applications.

The attractive styling and the large choice of interfaces make it the preferred choice for corporate environments. Advanced support of the widest range of technologies and exclusive security features allow high-end access control schemes to be deployed seamlessly.

SpringCard FunkyGate-DW NFC choice of interfaces include

- **RS-485**, with both a simple and an advanced protocol to address all needs,
- **Data+Clock (ISO2/Magstripe)**,
- **Wiegand (D0+D1)**.

Thanks to a **versatile Template System** (shared with all other **SpringCard** Readers and RFID/NFC Scanners), **SpringCard FunkyGate-DW NFC** is able to read either a serial number or virtually any data coming from standard ISO/IEC 14443 proximity cards, ISO/IEC 15693 vicinity labels or tags. It is also able to fetch NDEF data from RFID chips formatted according to one the NFC Forum Tag specifications, and to receive NDEF data from a NFC Forum “peer-to-peer” (SNEP server on top of LLCP).

This document provides all necessary information to configure the **FunkyGate-DW NFC** Reader and to develop a software that will handle data coming from the Reader, and to drive or re-configure the Reader when needed.

1.2. Supported product

Order code	Product
FPF13255	**FunkyGate-DW NFC:** new generation wall-mount RFID/NFC/contactless card Reader, with RS-485, Data+Clock and Wiegand interfaces

1.3. Audience

This manual is designed for use by application developers and system integrators. It assumes that the reader has a good knowledge of computer development and a good knowledge of the RFID/NFC technologies.

1.4. Support and updates

Useful related materials (product datasheets, application notes, sample software, HOWTOs and FAQs…) are available at SpringCard's web site:

www.springcard.com

Updated versions of this document and others are posted on this web site as soon as they are available.

For technical support enquiries, please refer to SpringCard support page, on the web at

www.springcard.com/support

1.5. Related documents

1.5.1. Product's specifications

You'll find the feature-list and the technical characteristics of every product in the corresponding leaflet.

Document ref.	Content
PFL13276	**FunkyGate-DW NFC** product leaflet

1.5.2. Common documentations

All SpringCard Readers and RFID Scanners products share the same "card processing" system through 1 to 4 processing templates. How the Reader process the card is therefore detailed in a document shared among all products in the family.

Document ref.	Content
PMA13205	**Readers / RFID Scanners Template System**

2. Hardware installation

To be written

3. Serial communication – Getting started

The Reader's serial port is able to operate into 3 modes:

- The MK1 serial protocol,
- The MK2 serial protocol,
- The Console mode.

Choosing between MK1 and MK2 protocols is decided by a Configuration Register (and therefore can't be changed until the Reader is reset).

The Console mode is entered at any time by sending

<ESC><ESC>shell<CR><LF>

as depicted later on.

Note that the SEC Configuration Register (h6E, § 10.7) may be used to disable the Console.

3.1. Physical layer

3.1.1. Electrical levels

The physical layer is compliant with the RS-485 standard.

3.1.2. Communication parameters

The default communication parameters are:

- Baudrate = 38400bps,
- 8 data bits,
- 1 stop bit,
- no parity,
- no flow control.

The baudrate could be changed by changing the Configuration Register SER ($_h$67, see § 10.4.1). The other parameters are fixed.

3.2. Validating your hardware integration

The easiest way to test your installation is to use a **terminal emulation software** running on a desktop or laptop computer. Popular terminal emulation software are **HyperTerminal** on Microsoft Windows, and **minicom** on Linux.

Of course, since the Reader uses RS-485 as physical layer, you shall use a RS-232 to RS-485 adapter, or a USB to RS-485 bridge, in order to connect to the Reader with the expected physical layer.

Open your terminal emulation software, set the communication parameter as specified above, and reset the Reader. You must see the Reader's startup string (§ 4.3.1).

Then hit the ESCAPE key twice, enter the string “info” (without the quotes), and hit the ENTER key. You must see the Reader's information data.

If one of those two tests fails, please double-check your hardware (wiring, power supply...) and the port number you've selected on the computer.

3.3. MK1 or MK2 protocol?

3.3.1. Which protocol shall you use?

The MK1 serial protocol is very simple and “human-readable”. This protocol is made for 1-to-1, peer-to-peer communication. It doesn't provide any kind of collision avoidance or collision detection feature, and therefore its reliability on a RS485 link is poor.

On the over hand, the MK2 protocol is a Master / Slave protocol designed for reliability and performance (collision avoidance, error detection and recovery, strict timings). This makes it the only choice every time more than one Reader must be connected to one Host.

3.3.2. Enabling MK1 protocol

To enable the MK1 protocol, assign the value $_h$0D to Configuration Register OPT ($_h$60) and h00 to Configuration Register SHD ($_h$68).

Using the Console (see below), this is done by sending

<ESC><ESC>cfg60=0D<CR><LF>

and

<ESC><ESC>cfg68=00<CR><LF>

3.3.3. Enabling MK2 protocol

To enable the MK2 protocol, assign the value $_h$0C to Configuration Register OPT ($_h$60).

Using the Console (see below), this is done by sending

<ESC><ESC>cfg60=0C<CR><LF>

The MK2 protocol needs an address for the Reader. This address must be written into Configuration Register SHD ($_h$68).

For instance, if the Address is $_h$17, using the Console (see below), this is done by sending

<ESC><ESC>cfg68=17<CR><LF>

Don't forget to assign a different address to every reader connected on the same RS-485 bus. All the Readers come out of factory with the address $_h$00. It is a good practice to assign only non-zero addresses, so the Host may accept a new Reader at any time and send a configuration command to the address $_h$00 assign a new address to this very Reader.

3.4. Reader's Console

The Reader features a “human” command processor (shell or console). This feature is primarily made for testing and demonstration purpose. Only the few commands depicted in this chapter could safely be used for configuration and diagnostic.

Note that the SEC Configuration Register (h6E, § 10.7) may be used to disable the Console.

3.4.1. Sending a Console command to the Reader

You must send the ASCII “Escape” character ($_b$1B) twice before being allowed to send a Console command to the Reader. In a terminal-emulation software, this is done by hitting the ESCAPE key twice.

Then write the command line as documented below, and terminate by hitting the ENTER key. Note that the Reader does not echo the entered characters; you should activate the local echo in your terminal-emulation software to see what you are typing.

The Reader accepts any end-of-line marker: <CR> alone, <LF> alone as well as <CR><LF> are valid.

3.4.2. List of Console commands

Command	Meaning
version	Show the firmware version
info	Show the firmware information data
show	Show the current configuration
cfg	Dump all Configuration Registers written into persistent memory
cfgXX=YY...YY	Write value $_h$YY...YY to Configuration Register $_h$XX
cfgXX=!!	Erase Configuration Register $_h$XX
cfgXX	Read Configuration Register $_h$XX
shell	Enable the Console (suppress the need to send ESCAPE twice before the commands). This is permanent until next reset, or until the **exit** command is invoked.
exit	Leave the Console (send ESCAPE twice to before next Console commands)
echo on	Turn echo ON
echo off	Turn echo OFF

4. MK1 serial protocol

4.1. Abstract

The MK1 serial protocol is very simple and “human-readable”: the Reader sends a frame every-time it “sees” a card, and the Host sends short commands whenever it wants to drive the Reader's LED or buzzer.

This protocol is made for 1-to-1, peer-to-peer communication. It doesn't provide any kind of collision avoidance or collision detection feature, and therefore its reliability on a RS485 link is poor. Prefer the MK2 serial protocol whenever it is possible to implement it in the host.

4.2. Physical layer

Please refer to § 3.1.

4.3. Reader → Host notifications

4.3.1. Startup string

When configured to use the MK1 serial protocol, upon reset, the Reader sends a startup string:

```
SpringCard S663/RDR x.xx
```

Where “x.xx” is the version number.

4.3.2. Notification when a card is read

When a card is discovered, the Template System (see chapter 11) is invoked and returns a small piece of data, which is the actual Card Identifier seen by the Reader (it could be either a serial number or some data coming from the card's internal memory – this depends on the Template involved).

The Card Identifier is transmitted by the reader over the serial link as follow:

<PREFIX><CARD IDENTIFIER><SUFFIX>

Normally (when the Template is well-configured for the given card), the Card Identifier is a sequence of numbers or letters according to the ASCII charset.

a. Default configuration

The default configuration is

- PREFIX = <BEL><STX>, where <BEL> is the ASCII “bell” or “ring” character ($_h$07) and <STX> the ASCII “Start of Text” character ($_h$02),
- SUFFIX = <ETX><CR><LF>, where <ETX> is the ASCII “End of Text” character ($_h$03), <CR> the ASCII “carriage return” character (h0D) and <LF> the ASCII “line feed” character ($_h$0A).

Therefore, the frame sent by the reader in its default configuration is

<BEL><STX><CARD IDENTIFIER><ETX><CR><LF>

b. Other configurations

Configuration Register SER ($_h$67) controls the format of the frame (see § 10.4.1)

4.3.3. Host's ACK

It is recommended that the Host sends the ASCII “Acknowledge” character ($_h$06) after receiving a valid Card Identifier.

<ACK>

4.4. Host → Reader commands (and Reader's ACK)

4.4.1. Command/response sequences

The Host may send any of the commands listed in § 4.4.3. The command frame has no prefix, and is terminated by <CR><LF>:

<COMMAND><CR><LF>

4.4.2. Reader's ACK

When a valid command is received from the Host, the Reader sends the ASCII “Acknowledge” character ($_h$06) within 50ms.

<ACK>

When an invalid command is received or a communication error occurs, the Reader sends the ASCII “Not Acknowledge” character ($_h$15) within 100ms after having detected the error.

<NAK>

4.4.3. List of commands

Command	Meaning
A0	Stop Reader (RF field OFF, no activity on the RF interface)
A1	Start Reader
R0	Red LED is switched OFF
R1	Red LED is switched ON
R2	Red LED blinks slowly
R3	Red LED blinks quickly
G0	Green LED is switched OFF
G1	Green LED is switched ON
G2	Green LED blinks slowly
G3	Green LED blinks quickly
Z0	Buzzer stops
Z1	Buzzer starts
Z2	Short buzzer sound
Z3	Long buzzer sound
C	Clear LED / buzzer (same as sending R0, G0, Z0)

5. MK2 serial protocol – Low layers

5.1. Abstract

The MK2 serial protocol is a Master / Slave protocol, the Host being the Master, and 1 to 255 Readers being the Slaves.

This protocol is designed with collision avoidance in mind: the Master queries every Slave one after the other, and a Slave is allowed to transmit only during the time-window opened by the Master.

5.2. Physical layer

Please refer to § 3.1.

5.3. Network layer

5.3.1. Block format

Every block transmitted over the physical link is formatted as follow:

STX	TYPE	ADDR	PAYLOAD	LRC	ETX
1 byte	1 byte	1 byte	Variable length	1 byte	1 byte

5.3.2. Description of the fields

Field	Description
STX	ASCII “Start of Text” character ($_h$02)
TYPE	The TYPE byte is used to convey the information required to control the data transmission. There are three fundamental types of blocks: • I-block used to convey information for use by the upper layers • R-block used to convey positive or negative acknowledgements • S-block used to exchange control information between the Master and the Slave
ADDR	The ADDR byte is used to identify a specific Reader, numbered between $_h$00 and $_h$FF • For Host → Reader blocks, this is the address of the target Reader, • For Reader → Host blocks, this is the address of the source Reader
PAYLOAD	The PAYLOAD field is optional, and could only be present within a I-Block. When present, the PAYLOAD field conveys application data.
LRC	The LRC byte if the exclusive OR (XOR) of all bytes from TYPE to PAYLOAD (i.e. STX and ETX not included)
ETX	ASCII “End of Text” character ($_h$03)

5.3.3. Escaping

This protocol has four reserved values that must be “escaped” (protected) if they appear in the block's content:

- The value for the ASCII “Start of Text” character (<STX>, $_h$02),
- The value for the ASCII “End of Text” character (<ETX>, $_h$03),
- The value for the ASCII “Escape” character (<ESC>, $_h$1B),
- The value for the ASCII “Data Link Escape” character (<DLE>, $_h$10).

If any of the TYPE, ADDR, PAYLOAD or LRC fields contains one of those reserved values, the corresponding byte must be “escaped” by sending <DLE> before the actual byte.

5.3.4. Size of the blocks

The size of every block must be less or equal to 69 bytes before escaping.

This lead to a PAYLOAD between 0 and 64 bytes.

If the application layer needs to transmit more than 64 bytes, chaining shall be used.

5.3.5. Format of the TYPE byte

a. I-Block

Bit	Description
7 (msb)	Direction • 0 for Host → Reader • 1 for Reader → Host
6	Shall be set to 0
5	Shall be set to 0
4	Chaining • 0: no chaining – this block is the only one, or the last one in a sequence • 1: chaining enabled – more block(s) to come
3	Block number, $_h$0 to $_h$F
2	
1	
0 (lsb)	

b. R-Block

Bit	Description
7 (msb)	Direction • 0 for Host → Reader • 1 for Reader → Host
6	Shall be set to 1
5	Block type: • $_b$00: R-OK (final acknowledgement) • $_b$01: R-ACK (positive acknowledgement) • $_b$10: R-NACK (negative acknowledgement) • $_b$11: RFU, do not use
4	
3	Block number, $_h$0 to $_h$F
2	
1	
0 (lsb)	

c. S-Block

Bit	Description
7 (msb)	Direction • 0 for Host → Reader • 1 for Reader → Host
6	Shall be set to 0
5	Shall be set to 1
4	Block type: • 0: S-WAIT • 1: S-ENUM
3	Block number, $_h$0 to $_h$F
2	
1	
0 (lsb)	

5.3.6. The ADDR byte

The Reader uses the address specified in byte 1 of its SHD Configuration Register ($_h$68, see § 10.4.2)

5.4. General communication flow

5.4.1. Definition of a sequence

A Sequence starts when the Host wants to send something to one Reader, or is willing-full to give one Reader to send something.

1. The Host sends one I-Block to the Reader (or more than one, see “Chaining” below). The block may be empty or may contain a Payload.
2. The Reader sends one I-Block to the Host in response (or more than one, see “Chaining” below). The block may be empty or may contain a Payload.
3. When receiving the last I-Block from the Reader, the Host sends one R-OK block to close the sequence.

5.4.2. Timings and S-WAIT block

The Reader ensures that it answer to every block coming from the Host by a response block within 75ms. The Host may use an 80ms-timeout to watch-out the Reader.

If the Reader is unable to provide a valid I-Block in the 75ms-time-window, the Reader sends an S-WAIT block.

Upon receiving an S-WAIT block, the Host knows that the Reader needs 500ms to achieve its pending tasks. During those 500ms, the Host may communicate with other Readers.

After 500ms or even more, the Host goes back to the first Reader by sending an I-block.

If the Reader is ready, it provides its actual answer in an I-Block within 75ms. Otherwise, the Reader may send a new S-WAIT block to beg for 500ms more.

5.4.3. Chaining

If the application data buffer is longer than the max size for the PAYLOAD field, the data shall be divided onto multiple I-Blocks. In this case, the Chaining bit is set to 1 for every I-Block but the last one.

Every I-Block with the Chaining bit set to 1 shall be acknowledged by an R-ACK block.

Chaining is not implemented in the current version of the Reader's firmware. The Host shall not use this feature (and the Reader will not use it).

5.4.4. Block numbering rules

The Reader must answer to every block received from the Host by a block having the same block number.

The Host is free to change its block number sequentially or randomly to detect communication errors.

5.5. Error handling and recovery

5.5.1. For the Reader

When any error is detected by the Reader, the Reader remains quiet. It is up to the Host to try to restore the communication link.

5.5.2. For the Host

- **Invalid block number:** is the Host receives a block that contains a block number not equal to the last one sent, it shall send an R-NACK block with the last block number to make the Reader repeat its last block,
- **LRC error:** is the Host receives a block that contains an invalid LRC, it shall send an R-NACK block with the last block number to make the Reader repeat its last block,
- **Timeout error:** if the Reader doesn't answer in the give time-window, the Host shall repeat its last block (with the same block number). If the Reader remains mute, the Host may permanently ignore this Reader,
- **Protocol error:** if the Host received a well-formed yet invalid block from a Reader, the Host may permanently ignore this Reader.

5.6. Initial enumeration

The Host may send S-ENUM blocks to all addresses to detect the installed Readers. When receiving an S-ENUM block, the Reader answers by an S-ENUM block within 4ms only.

> *Note: it is also possible to enumerate the Readers using only I-Block queries, but the time-window opened for an I-Block is 80ms, which makes the enumeration using S-ENUM blocks lots faster.*

6. MK2 serial protocol – Application layer

6.1. Principles

The application-level communication uses the T,L,V scheme:

- **T (Tag):** this is the operation-code of a command, or the identifier of a data field. The Tag is on either 1 or 2 bytes,
- **L (Length)**: this is the length of the following Value, on 1 byte. Allowed values are $_h$00 to $_h$7F,
- **V (Value):** the parameters to the command, or the data field itself. The length is specified by L, from 0 to 127 bytes.

6.2. Host → Reader

6.2.1. Get Global Status

T	L
$_h$00	$_h$00

The Reader answers by 2 frames:

1. Reader Identifier
2. Tamper Status

6.2.2. Start/Stop Reader

T	L	V
$_h$0A	$_h$01	**mode**

- **mode:** start/stop command
  - $_h$00 Reader goes OFF (RF field OFF, no activity on RF)
  - $_h$01 Reader goes ON

6.2.3. Clear LEDs command

Both red and green LEDs go OFF.

T	L
$_h$D000	$_h$00

6.2.4. Set LEDs command

Both red and green LEDs are driven – until a Clear LEDs command is received.

T	L	V	
$_h$D000	$_h$02	red	green

- **red:** command for red LED
  - $_h$00 OFF
  - $_h$01 ON
  - $_h$02 blinks slowly
  - $_h$03 blinks quickly
- **green:** command for green LED
  - $_h$00 OFF
  - $_h$01 ON
  - $_h$02 blinks slowly
  - $_h$03 blinks quickly

6.2.5. Start LED sequence command

Both red and green LEDs are driven – until a Clear LEDs command is received or a timeout occurs.

T	L	V		
$_h$D000	$_h$04	red	green	time (sec)

- **red:** same as above,
- **green:** same as above,
- **time:** time (in seconds, MSB-first) before returning to all-LED-OFF state.

6.2.6. Buzzer command

T	L	V
$_h$D100	$_h$01	seq.

- **seq:**
  - $_h$00 buzzer OFF,
  - $_h$01 buzzer ON,
  - $_h$02 buzzer short sequence,
  - $_h$03 buzzer long sequence.

6.3. Reader → Host

6.3.1. Reader Identifier

This T,L,V is transmitted in response to the **Get Global Status** command.

T	L	V
$_h$8100	$_h$1C	`SpringCard S663/RDR x.xx`

6.3.2. Tamper Status

This T,L,V is transmitted in response to the **Get Global Status** command or when one of the tampers is broken/restored.

T	L	V
$_h$2F	$_h$01	Bit field, the broken tampers are denoted by the corresponding bit set to 1. V = $_h$00 when all tampers are OK.

6.3.3. Card Identifier

This T,L,V is transmitted when the Reader has read a card.

T	L	V
$_h$B000	<var.>	Card Identifier

6.3.4. Card removed

This T,L,V is transmitted when the card is removed, if the Reader is configured to do so (OPT Configuration Register, $_h$60, §10.1).

T	L
$_h$B000	$_h$00

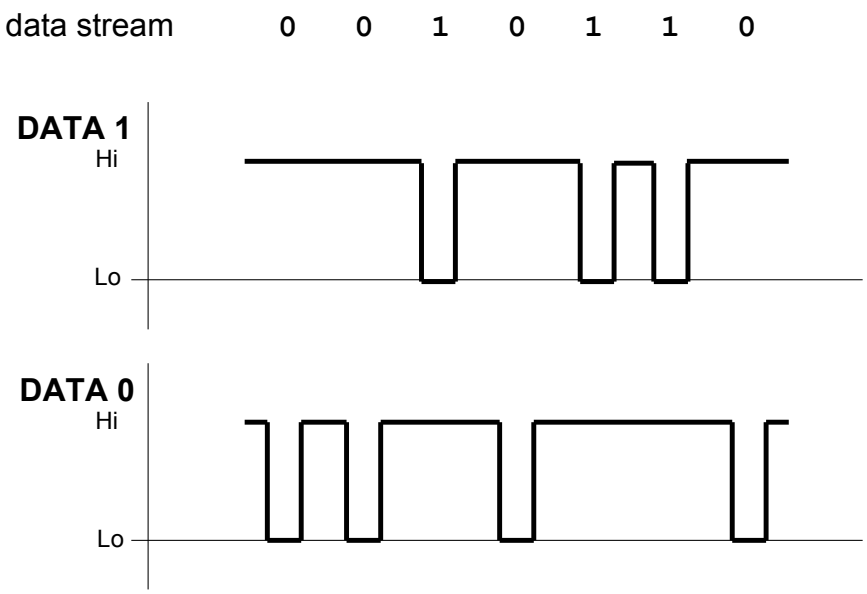
7. Wiegand output

7.1. The Wiegand interface

7.1.1. Principles

The Wiegand interface has 2 lines, named D0 and D1. Both lines are active low (i.e. at high level when idle).

- A falling edge on line D0 denotes bit 0,
- A falling edge on line D1 denotes bit 1.



Due to the lack of return line, the Wiegand interface is strictly Reader → Host. Anyway, the Reader provides two input lines to control its red and green LEDs.

7.1.2. Electrical levels

D0 and D1 are open-collector output lines. Pull-up resistors must be provided on the Host side.

Recommended values are

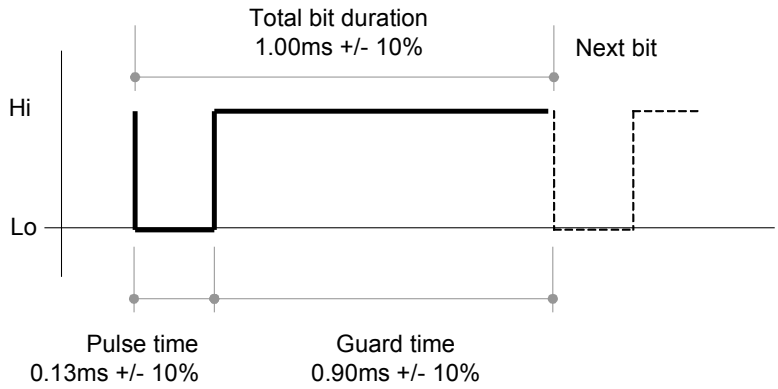
- $R_{pull-up}$ = 4.7kΩ if pulled to 5V (giving I_{max} in the D0 or D1 lines = 1.0mA),
- $R_{pull-up}$ = 10kΩ if pulled to 12V (giving I_{max} in the D0 or D1 lines = 1.2mA).

Keep pull-up voltage below 15V and choose $R_{pull-up}$ to ensure that I_{max} remains under 10mA.

The Host shall trigger the falling edge of both signals (since the rising edge is generally less clean due to the open collector / pull-up hardware).

7.1.3. Timings

a. Default configuration



b. Changing the timings

To change the timings, edit the WGD Configuration Register ($_h$65, see § 10.5).

7.2. Reader → Host notifications

When a card is discovered, the Template System (see § 11) is invoked and returns a small piece of data, which is the actual Card Identifier seen by the Reader (it could be either a serial number or some data coming from the card's internal memory – this depends on the Template involved).

The Card Identifier is transmitted by the reader over the Wiegand link “as is”:

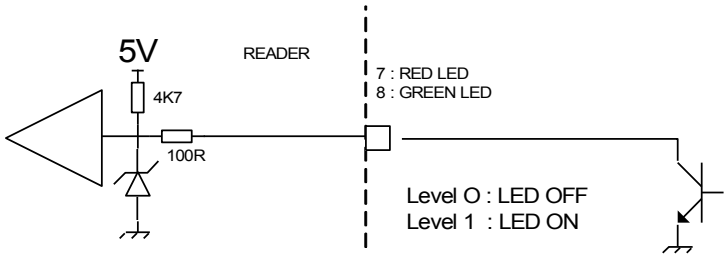
<CARD IDENTIFIER>

Please review carefully the configuration of the templates you're using, to make sure all of them return an identifier that makes senses for your Host's Wiegand input.

7.3. Driving the Reader's LEDs

The Reader features 2 input lines, called LR and LG, to drive the red LED and the green LED respectively. Both lines are pulled-up to 5V inside the Reader, by the mean of a 4.7kΩ resistor.

The Host shall implement open-collector output lines. Tying either line to low level (below 1.7 V) switches the corresponding LED ON.



8. Data+Clock/ISO2/Magstripe output

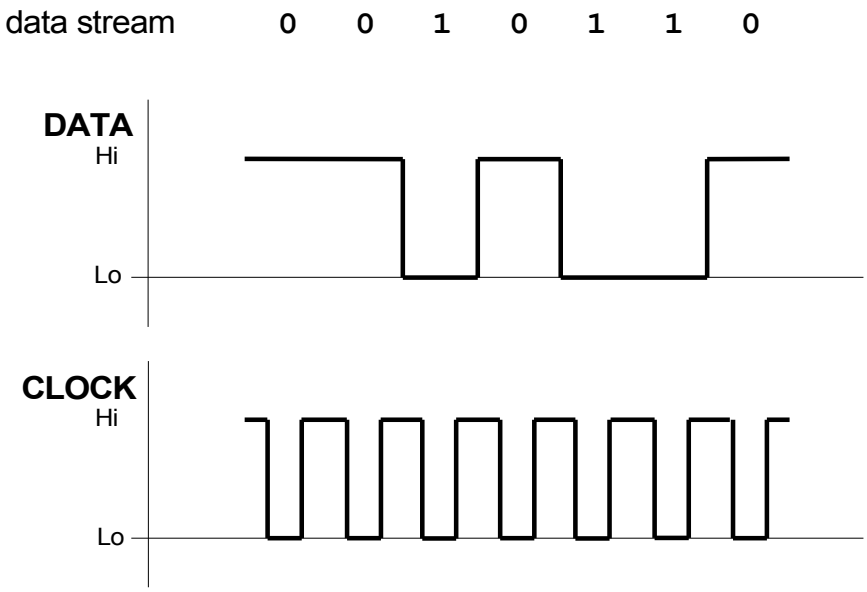
8.1. The Data+Clock interface

8.1.1. Principles

The Data+Clock interface has 2 lines, named DATA and CLOCK. Both lines are active low (i.e. at high level when idle).

The Reader is the master of an inverting, synchronous serial communication scheme:

- DATA high → means bit value = 0,
- DATA low → means bit value = 1,
- DATA shall be sampled by the Host on the falling edge of CLOCK.



Due to the lack of return line, the Data+Clock interface is strictly Reader → Host. Anyway, the Reader provides two input lines to control its red and green LEDs.

8.1.2. Electrical levels

DATA and CLOCK are open-collector output lines. Pull-up resistors must be provided on the Host side.

Recommended values are

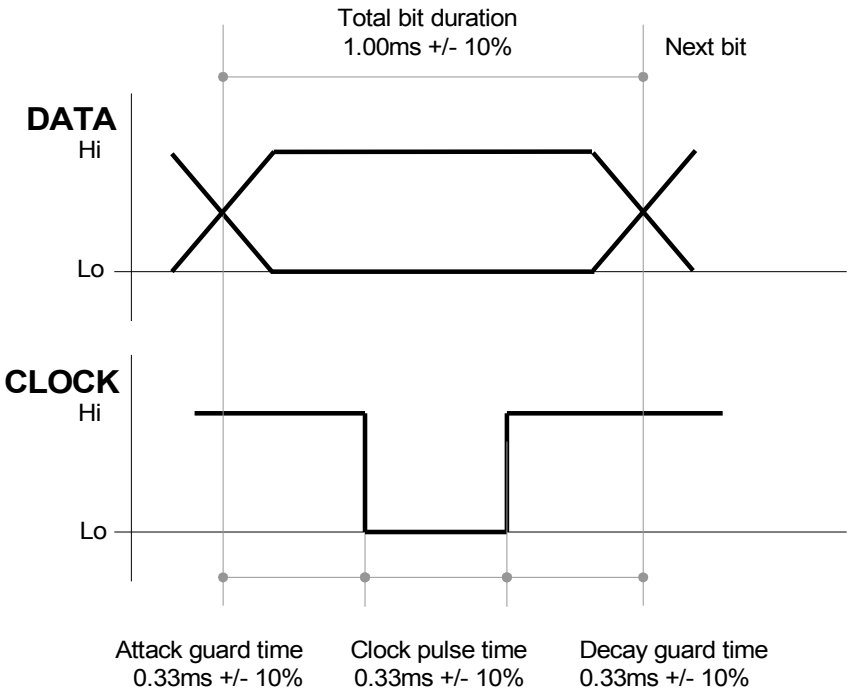
- $R_{pull-up}$ = 4.7kΩ if pulled to 5V (giving I_{max} in the CLOCK or DATA lines = 1.0mA),
- $R_{pull-up}$ = 10kΩ if pulled to 12V (giving I_{max} in the CLOCK or DATA lines = 1.2mA).

Keep pull-up voltage below 15V and choose $R_{pull-up}$ to ensure that I_{max} remains under 10mA.

The Host shall trigger the falling edge of the CLOCK signal (since the rising edge is generally less clean due to the open collector / pull-up hardware).

8.1.3. Timings

a. Default configuration



b. Changing the timings

To change the timings, edit the DTC Configuration Register ($_h$66, see § 10.6).

8.2. ISO2/Magstripe protocol

"ISO2/Magstripe" is a subset of the general "Data+Clock" scheme. According to this protocol,

- Only decimal (0-9) digits could be transmitted,
- Every digits is transmitted over 5 bits: the 4-bit BCD value, lsb-first, followed by an odd parity bit,
- Every frame starts by a synchronisation sequence, and includes a LRC,
- Values $_d$10-$_d$15 ($_h$A-$_h$F) are used for signalling, or are RFU.

8.3. Reader → Host notifications

When a card is discovered, the Template System (see § 11) is invoked and returns a small piece of data, which is the actual Card Identifier seen by the Reader (it could be either a serial number or some data coming from the card's internal memory – this depends on the Template involved).

The Card Identifier is transmitted by the reader over the Data+Clock link after embedding into a valid ISO2/Magstripe frame:

<SYNCHRO><START><CARD IDENTIFIER><STOP><LRC><SYNCHRO>

To ensure the frame is compliant with ISO2/Magstripe protocol, the Card Identifier shall contain only decimal (BCD) values.

Please review carefully the configuration of the templates you're using, to make sure all of them return a Card Identifier that is purely decimal.

8.3.1. "Synchro" sequence

The synchronisation sequence is made of 16 bits equal to 0.

8.3.2. "Start" symbol

The start symbol is the nibble $_h$B.

Symbol	Hex. value	Bit pattern (including parity)
START	$_h$B	$_b$1101 0

8.3.3. "Stop" symbol

The stop symbol is the nibble $_{h}$F.

Symbol	Hex. value	Bit pattern (including parity)
STOP	$_{h}$F	$_{b}$1111 1

8.3.4. Data

Symbol	Hex. value	Bit pattern (including parity)
0	$_{h}$0	$_{b}$0000 1
1	$_{h}$1	$_{b}$1000 0
2	$_{h}$2	$_{b}$0100 0
3	$_{h}$3	$_{b}$1100 1
4	$_{h}$4	$_{b}$0010 0
5	$_{h}$5	$_{b}$1010 1
6	$_{h}$6	$_{b}$0110 1
7	$_{h}$7	$_{b}$1110 0
8	$_{h}$8	$_{b}$0001 0
9	$_{h}$9	$_{b}$1001 1

Depending on the template, a separator character could be transmitted.

Symbol	Hex. value	Bit pattern (including parity)
SEPARATOR	$_{h}$D	$_{b}$1011 0

8.3.5. LRC

The LRC is the exclusive OR (XOR) of <START><CARD IDENTIFIER><STOP> on 4 bits.

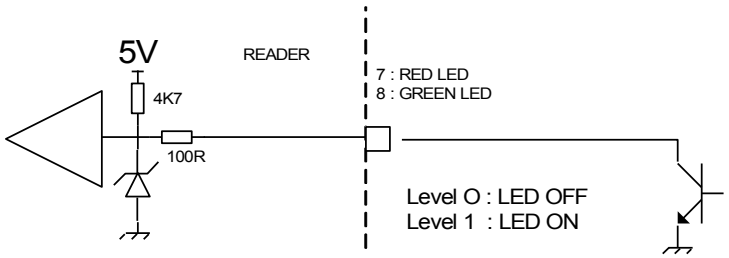
The LRC uses the same odd parity scheme as the data nibbles.

Value	Hex. value	Bit pattern (including parity)
0	$_h$0	$_b$0000 1
1	$_h$1	$_b$1000 0
2	$_h$2	$_b$0100 0
3	$_h$3	$_b$1100 1
4	$_h$4	$_b$0010 0
5	$_h$5	$_b$1010 1
6	$_h$6	$_b$0110 1
7	$_h$7	$_b$1110 0
8	$_h$8	$_b$0001 0
9	$_h$9	$_b$1001 1
10	$_h$A	$_b$0101 1
11	$_h$B	$_b$1101 0
12	$_h$C	$_b$0011 1
13	$_h$D	$_b$1011 0
14	$_h$E	$_b$0111 0
15	$_h$F	$_b$1111 1

8.4. Driving the Reader's LEDs

The Reader features 2 input lines, called LR and LG, to drive the red LED and the green LED respectively. Both lines are pulled-up to 5V inside the Reader, by the mean of a 4.7kΩ resistor.

The Host shall implement open-collector output lines. Tying either line to low level (below 1.7 V) switches the corresponding LED ON.



9. Editing Reader's configuration

The Reader's configuration is stored in a set of non-volatile Configuration Registers. There are two groups of Registers:

- The Registers that control the behaviour of the Reader are fully documented in chapter 10. Some of them are common to various SpringCard Readers, but some of them are very specific to the **SpringCard FunkyGate-DW NFC**.
- The Registers that control the Template System are shared among all SpringCard Readers. Chapter 11 is therefore a place-holder that redirects to the document describing this Template System precisely.

But this subtle distinction between these two groups is only there to keep the documents short, and to ease switching from one Reader to the other. Technically speaking, all Registers are defined (and accessed) the same way.

There are two ways to edit the Reader's Configuration Registers:

1. Through the serial link, using the Console
2. Using Master Cards
3. Using a NFC mobile phone

> *Note that the SEC Configuration Register ($_h$6E, § 10.7) may be used to disable either way to access the Configuration Registers.*

9.1. Through the serial link

Enter the Console by sending **<ESC><ESC>shell<CR><LF>** as instructed in § 3.4.

9.1.1. Reading Configuration Registers

Enter “cfg” to list all Configuration registers currently defined (registers that are not explicitely defined keep their default value).

Enter “cfgXX” to read the value of the Configuration register $_h$XX.

Note that Configuration registers $_h$55, $_h$56 and $_h$6F that hold sensitive data (the keys used by Master Cards and the Reader's pin-code) are masked.

9.1.2. Writing Configuration Registers

Enter “cfgXX=YYYY” to update Configuration Register $_h$XX with value $_h$YYYY. YYYY can me any length between 1 and 32 bytes.

Enter “cfgXX=!!” to erase Configuration Register $_h$XX.

9.2. Using Master Cards

Preliminary information – not available yet.

The Master Cards are NXP Desfire cards formatted and programmed by **SpringCard Configuration Tool (ScMultiConf.exe, ref # SN14007)** for Windows.

Please refer to this software's documentation for details.

9.3. Using a NFC mobile phone

Preliminary information – not available yet.

10. Global configuration of the Reader

10.1. General options

Name	Tag	Description	Size
OPT	h60	General option, see table below	1 or 2

General options bits

Bits	Value	Meaning
		Byte 0
7	0	Normal mode
	1	Power saving mode (the Reader is slower)
6	0	Track the cards by their ID only
	1	Keep the RF field active to track the cards (works with Random IDs)
5 - 4		***Anti-collision mode***
	00	Read every card one after the other
	01	*RFU*
	10	Read only one card at a time (ignore the other ones)
	11	Prevent reading when there's more than one card in the field
3 - 2		***Master Card and NFC configuration***
	00	Disable configuration by Master Card or NFC
	01	Allow configuration by Master Card or NFC at power up only
	10	*RFU*
	11	Allow configuration by Master Card or NFC all the time
1 - 0		***Output interface***
	00	RS485, MK2 protocol
	01	RS485, MK1 protocol
	10	Wiegand
	11	Data+Clock
		Byte 1 (optional)
7	0	Do not send Card removed notification
	1	Send Card removed notification (§ 6.3.4)
6	0	*RFU (set to 0)*
5 - 4		***MK2 link timeout***
	00	1 s
	01	3 s
	10	10 s
	11	30 s
3	0	*RFU (set to 0)*

2	0	*RFU (set to 0)*
1	0	*RFU (set to 0)*
0	0	Reader is active on startup
	1	Reader is not active on startup (Host must send an activation command)

Default value: $_b$00001101 00000000 (MK1 protocol)

10.2. Delays and repeat

Name	Tag	Description	Min	Max
ODL	$_h$61	Min. delay between 2 consecutive outputs (0.1s)	0	100
RDL	$_h$62	Min. delay between 2 consecutive identical outputs (0.1s) A value of 255 means that the card must be removed from the field –and re-inserted into– before being read again	0	100

Default value: ODL = 5 (1ms) RDL = 20 (2s)

10.3. LEDs and Buzzer

Name	Tag	Description	Size
CLD	$_h$63	LEDs control, see table below	1
CBZ	$_h$64	Buzzer control, see table below	1

LEDs control bits

Bits	Value	Meaning
7	0	Short LED sequences (3 seconds)
	1	Long LED sequences (10 seconds)
6 - 5	00	When idle, blue LED blinks slowly (“heart beat” sequence)
	01	When idle, blue LED is always on
	10	When idle, blue LED is always off
	11	RFU
4	0	Green LED stays OFF
	1	Green LED blinks when a valid card has been processed
3	0	Red LED stays OFF
	1	Red LED blinks when an unsupported card has been processed
2	0	Green LED stays OFF
	1	Green LED blinks as soon as a card is seen in the field
1 - 0	00	Data+Clock or Wiegand: LED driven by input lines (active low)
	01	RS-485: LED driven by Host commands only
	10	Data+Clock or Wiegand: LED driven by input lines (active high)
	11	RS-485: LED driven by internal state machine and Host commands

Default value: $_b$00001111

Buzzer control bits

Bits	Value	Meaning
7	0	Buzzer short pulse = 0,2 sec
	1	Buzzer short pulse = 0,5 sec
6	0	Buzzer long pulse = 0,7 sec
	1	Buzzer long pulse = 1,5 sec
5		*RFU*
4	0	No action on buzzer before specified by host controller
	1	Short pulse when a valid card has been processed
3	0	No action on buzzer for unsupported cards
	1	Long pulse when an unsupported card has been processed
2	0	No action on buzzer before processing is achieved
	1	Short pulse as soon as a card is seen in the field
1 - 0	00	Buzzer is disabled, other settings are ignored
	01	Buzzer controlled by serial commands, other settings are ignored
	10	Buzzer controlled by internal software, serial commands are ignored
	11	Buzzer controlled by both internal software and serial commands

Default value : $_{b}$00010010

10.4. Serial communication

10.4.1. Baudrate and frame format for MK1 protocol

Name	Tag	Description	Size
SER	$_h$67	Serial configuration bits. See table a below	1

Serial configuration bits

Bits	Value	Meaning
7	0	No STX / ETX frame markers
	1	Use STX and ETX as frame markers
6 - 5	00	No BEL / TAB / CR/LF frame markers
	01	Use CR/LF only
	10	Use BEL and CR/LF as frame markers
	11	Use TAB and CR/LF as frame markers
4 - 3		**Serial Repeat**
	00	Dot no repeat
	01	Repeat 4 times with timeout of 100ms (Host must send an ACK to cancel)
	10	Repeat 4 times with timeout of 250ms (Host must send an ACK to cancel)
	11	Repeat 9 times with timeout of 250ms (Host must send an ACK to cancel)
2 - 0		**Baudrate**
	000	1200bps
	001	2400bps
	010	4800bps
	011	9600bps
	100	19200bps
	101	38400bps
	110	RFU
	111	115200bps

Default value: $_b$11000101

NB: bits 7-3 are ignored when the MK2 protocol is selected, but the Baudrate configuration is shared among both protocols (and among the Console too).

10.4.2. Addressing

Name	Tag	Description	Size
SHD	$_h$68	Reader address	1

RS-485 configuration bits

Byte	Value	Meaning
7 - 0	$_h$00 to $_h$FF	Address of the Reader on the bus is MK2 protocol is selected. This value must be $_h$00 when the MK1 protocol is selected.

Default value: $_b$00000000 (empty address for MK1 protocol)

10.5. Wiegand

Name	Tag	Description	Size
WGD	$_h$65	Wiegand configuration bits. See table **a** below.	1

Wiegand configuration bits

Byte	Value	Meaning
		Wiegand output options
7 - 4	0000	"as is" (no parity, no LRC)
	0001	*RFU*
	0010	*RFU*
	0011	*RFU*
	0100	*RFU*
	0101	*RFU*
	0110	*RFU*
	0111	*RFU*
	1000	*RFU*
	1001	*RFU*
	1010	*RFU*
	1011	*RFU*
	1100	Add 2+1 parity bits
	1101	R*FU*
	1110	*RFU*
	1111	*RFU*
		Wiegand timings : guard time
3 - 2	00	guard time = 250µs
	01	guard time = 1000µs
	10	guard time = 1500µs
	11	guard time = 3000µs
		Wiegand timings : pulse time
1 - 0	00	pulse time = 25µs
	01	pulse time = 50µs
	10	pulse time = 100µs
	11	pulse time = 200µs

Default value : $_b$00001010

10.6. Data+Clock

Name	Tag	Description	Size
DTC	$_h$66	Dataclock configuration bits. See table **a** below.	1

Dataclock configuration bits

Byte	Value	Meaning
7	0	Standard ISO2 / Magstripe frame[1]
	1	*Raw output (bits 3-2 are ignored)*[2]
6 - 4		*RFU*
3 - 2		**Dataclock output options**
	00	Non-decimal digits in the output frame are discarded
	01	Non decimal digits in the output frame are replaced by separators
	10	Dataclock translation method 1
	11	Dataclock translation method 2
1 - 0		**Dataclock timing**
	00	clock pulse = 100µs
	01	clock pulse = 200µs
	10	clock pulse = 330µs
	11	clock pulse = 500µs

Default value : $_b$00000010

[1] Frame starts with 0xB, ends with 0xF + 4 bits LRC. Only decimal digits can be transmitted as 4-bit nibbles. A parity bit is transmitted with each nibble.

[2] No frame marker, no LRC, no parity bits.

10.7. Security options

Name	Tag	Description	Size
SEC	$_h$6E	Security option bits. See table **a** below.	1

Security option bits

Bits	Value	Meaning
7	1	*RFU (set to 1)*
6	0	*RFU (set to 0)*
5	0	*RFU (set to 0)*
4	0	*RFU (set to 0)*
3	0	*RFU (set to 0)*
Tampers		
2	0	Do not signal tamper alarms on buzzer
	1	Signal tamper alarms on buzzer
1	0	Reader keeps on reading even if a tamper is broken
	1	Reader stops reading when a tamper is broken
0	0	Do not raise alarm if a tamper is broken at power up
	1	Raise alarm on tamper broken even at power up

Default value: $_b$10000100

10.8. PIN code

Name	Tag	Description	Size
PIN	$_h$6F	PIN code to access reader's console.	2

Default value : empty (no pin-code)

Use this tag to define a 4 digits PIN code to protect access to reader's console.

The 2-byte value must store 4 valid BCD digits, or the reserved value $_h$FFFF that permanently disables the console feature.

11. The Template System

SpringCard FunkyGate-DW NFC provides 4 “Card Processing Templates” that defines how the Reader which fetch data from various cards/tags, and how the Card Identifier will be constructed from these data before being sent to the Host.

The template system is fully described in document # **PMA13205 “Readers / RFID Scanners Template System**”.

Please use this document as reference to configure the “Reader part” of your **SpringCard FunkyGate-DW NFC.**

12. 3rd-party licenses

SpringCard FunkyGate-DW NFC has been developed using open-source software components.

12.1. FreeRTOS

FreeRTOS is a market leading real time operating system (or RTOS) from Real Time Engineers Ltd. **SpringCard FunkyGate-DW NFC** runs on FreeRTOS v7.5.2.

FreeRTOS is distributed under a modified GNU General Public License (GPL) that allows to use it in commercial, closed-source products.

For more information, or to download the source code of FreeRTOS, please visit

www.freertos.org

Disclaimer

This document is provided for informational purposes only and shall not be construed as a commercial offer, a license, an advisory, fiduciary or professional relationship between PRO ACTIVE and you. No information provided in this document shall be considered a substitute for your independent investigation.

The information provided in document may be related to products or services that are not available in your country.

This document is provided "as is" and without warranty of any kind to the extent allowed by the applicable law. While PRO ACTIVE will use reasonable efforts to provide reliable information, we don't warrant that this document is free of inaccuracies, errors and/or omissions, or that its content is appropriate for your particular use or up to date. PRO ACTIVE reserves the right to change the information at any time without notice.

PRO ACTIVE doesn't warrant any results derived from the use of the products described in this document. PRO ACTIVE will not be liable for any indirect, consequential or incidental damages, including but not limited to lost profits or revenues, business interruption, loss of data arising out of or in connection with the use, inability to use or reliance on any product (either hardware or software) described in this document.

These products are not designed for use in life support appliances, devices, or systems where malfunction of these product may result in personal injury. PRO ACTIVE customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify PRO ACTIVE for any damages resulting from such improper use or sale.

Copyright notice

Editor's information

PRO ACTIVE SAS company with a capital of 227 000 €

RCS EVRY B 429 665 482

Parc Gutenberg, 2 voie La Cardon

91120 Palaiseau – FRANCE

Contact information

For more information and to locate our sales office or distributor in your country or area, please visit

www.springcard.com