IWM-X contactless reader

Reference manual

PMA8P3P revision AA
15/01/2009

TABLE OF CONTENT

1. INTRODUCTION

This document provides detailed technical information for use of the SpringCard wall-mount contactless proximity card reader IWM-X.

1.1. AUDIENCE

This reference manual assumes that the reader has expert knowledge of electronics. It is designed to be used by system integrators.

1.2. PRODUCT BRIEF

a. *Abstract*

IWM-X is a wall-mount proximity reader. It reads serial number or data from any standard ISO/IEC 14443 contactless card, including popular NXP MIFARE and DESFire families, and also ISO/IEC 15693 vicinity tags used in RFID systems.

IWM-X is 100% compliant with earlier IWM-K632.

b. *Typical applications*

This reader is primarily dedicated to corporate access control, where a high level of security or versatility is needed, but can also be used in cash or vending machines.

c. *Output modes*

Depending on software configuration (stored in non-volatile memory), the same reader can be operated into 3 modes :

- Wiegand (output only), with configurable frame length,
- Dataclock or ISO2 / Magstripe (output only),
- Serial input/output.

Depending on the underlying hardware, the serial input/output can either be RS-232, RS-485, or USB (USB to serial bridge).

d. *On the field configuration*

IWM-X is fully configurable on-the-field through secured Master Cards. Internal MD5, DES and 3-DES cryptographic algorithms are available for advanced security operations.

1.3. RELATED DOCUMENTS

You'll find any details regarding hardware and physical characteristics of each reader in the corresponding datasheet.

Datasheet	Covered products

2. CONFIGURATION DATA

There are two families of data :

- Global settings,
- Card Processing Templates.

Global settings specify output format and timings.

Card Processing Templates specify which kind of cards shall be read (ISO/IEC 14443, Mifare, Desfire, T=CL), how they must be read (serial number, data in file, …), and how the operation is secured (Mifare authentication, Desfire 3-DES secure session, …).

IWM-X can run 1 to 4 Card Processing Template simultaneously (+ 1 for Master Cards). This means that 4 different kinds of cards can coexist on a single site and can be read by a single IWM-X reader.

a. Configuration tags

Each configuration data is recognized by its "tag" and its length. The tag is a one-byte value, that uniquely identify the data.

The list of available tags, and their meaning, is the purpose of this chapter.

> ✋ Unless specified, each configuration data is exactly one byte (8 bits) long.

b. Non-volatile memory endurance

IWM-X configuration data are stored in reader's non-volatile memory (flash). They can be changed up to 100 times.

> ☠ Changing the configuration settings more than 100 times may permanently damage your IWM-X reader.

2.2. GLOBAL SETTINGS

The following tables enumerate all the data made available when configuring the reader.

2.2.1. General options

Name	Tag	Description	Size
OPT	${}_{h}60$	General options. See table **a** below.	1

a. *General options bits*

Bit	Value	Meaning
7	0 1	Normal mode Power saving mode[1]
6	0 1	Shutdown RF field when idle Shutdown RF field only when no card detected[2]
5 – 4	 00 01 10 11	***Anti-collision model :*** Process every card one after the other *RFU* When 2 cards are in the field, process the 1st and ignore the 2nd When 2 cards are in the field, ignore both
3 – 2	 00 01 10 11	***Master Card :*** Master Cards are disabled[3] Master Cards are enabled at power up *RFU* Master Cards are enabled all the time
1 – 0	 00 01 10 11	***Output interface :*** serial duplex (RS-232, USB) reader serial half-duplex (RS-485) reader Wiegand reader Dataclock reader

Default value : ${}_{b}10001101$

(power saving mode, Master Cards are enabled all the time, RS-485)

[1] When this value is selected, the card detection loop runs only every 250ms. In the meantime, RC chipset is OFF to reduce average power consumption. Do not choose this mode if you need fast operation at the gates, since it will increase transaction time at least by 250ms.

[2] This is required if strict anti-collision (bits 5-4 = ${}_{b}10$ or ${}_{b}11$) is needed.

[3] Configuration settings can only be altered through serial link

2.2.2. Delays and repeat options

Name	Tag	Description	Min	Max
ODL	h61	Min. delay between 2 consecutive outputs (0.1s).	0	100
RDL	h62	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 = 2 (200ms) RDL = 10 (1s)

2.2.3. LED and buzzer control options

Name	Tag	Description	Size
CLD	h63	LEDs control. See table **a** below.	1
CBZ	h64	Buzzer control. See table **b** below.	1

a. LEDs control bits

Bit	Value	Meaning
7	0 1	Short LED sequences (3 seconds) Long LED sequences (10 seconds)
6 – 5	00 01 10 11	When idle, blue LEDs blinks slowly (“heart beat” sequence) When idle, blue LEDs is always on When idle, blue LEDs is always off *RFU*
4	0 1	No action on green LED before specified by host controller Green LED blinks when a valid card has been processed
3	0 1	No action on red LED for unsupported cards Red LED blinks when an unsupported card has been processed
2	0 1	No action on green LED before processing is achieved Green LED blinks as soon as a card is seen in the field
1 – 0	00 01 10 11	LED control by hardware lines, other settings are ignored LED control by serial commands, other settings are ignored *RFU* LED control by internal software and serial commands

Default value : b00001111

b. *Buzzer control bits*

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

Default value : $_b$00010011

2.2.4. Wiegand mode

Name	Tag	Description	Size
WGD	h65	Wiegand configuration bits. See table **a** below.	1

a. Wiegand configuration bits

Bit	Value	Meaning
7 – 4		**Wiegand output options :**
	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	*RFU*
	1110	*RFU*
	1111	*RFU*
3 – 2	00	Wiegand guard time = 250µs
	01	Wiegand guard time = 1000µs
	10	Wiegand guard time = 1500µs
	11	Wiegand guard time = 3000µs
1 – 0	00	Wiegand pulse time = 25µs
	01	Wiegand pulse time = 50µs
	10	Wiegand pulse time = 100µs
	11	Wiegand pulse time = 200µs

Default value : $_b$00001010

See chapter 5.1 for details on Wiegand timings.

2.2.5. Dataclock mode

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

a. Dataclock configuration bits

Bit	Value	Meaning
7	0 1	Standard ISO2 / Magstripe frame[4] Raw output (bits 3-2 are ignored)[5]
6 – 4		*RFU*
3 – 2	 00 01 10 11	*See chapter **Dataclock App. Note** for details* Non-decimal digits in the output frame are discarded Non decimal digits in the output frame are replaced by separators Dataclock translation method 1 Dataclock translation method 2
1 – 0	00 01 10 11	Dataclock clock pulse = 100µs Dataclock clock pulse = 200µs Dataclock clock pulse = 330µs Dataclock clock pulse = 500µs

Default value : $_b$00000010

See chapter 6.2 for details on Dataclock timings.

[4] 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.

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

2.2.6. *Serial mode (RS-485, RS-232, USB)*

Name	Tag	Description	Size
SER	$_{h}$67	Serial configuration bits. See table **a** below.	1

a. *Serial configuration bits*

Bit	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 marker
	01	Use CR/LF as end of frame
	10	Use BEL as start of frame and CR/LF as end of frame
	11	Use TAB as start of frame and CR/LF as end of frame
4 – 3		*RFU*
		Baudrate :
2 – 0	000	1200bps
	001	2400bps
	010	4800bps
	011	9600bps
	100	19200bps
	101	38400bps
	110	*RFU*
	111	115200bps

Default value : $_{b}$11000101

☠ The baudrate parameter is common to USB, RS-232 and RS-485 interfaces.

Even if it is allowed, do not set baudrate to 115200bps when working with RS-485 interface, as the hardware and the characteristics of the bus aren't able to support it.

b. *Serial frame format*

Serial frames are always transmitted using ASCII representation of binary values.

For example, data '00 7A 12 6C 59 F4 04' (hexadecimal notation) are transmitted as string "007A126C59F404".

c. *Serial frame markers*

Bits 7-5 drive the start of frame / end of frame markers.

See chapter **Serial App. Note** for details.

2.2.7. RS-485 mode

Name	Tag	Description	Size
SHD	$_{h}68$	RS-485 configuration bits. See table **a** below.	1

a. RS-485 configuration bits

Bit	Value	Meaning
7 – 4		*RFU*
3 – 0	0000	Addressing disabled (single device on bus)
	0001 to 1110	Address = $_{h}01$ ($_{d}1$) to address = $_{h}0E$ ($_{d}14$)
	1111	*RFU*

2.2.8. Keep-alive & Break

Name	Tag	Description	Size
KAL	$_{h}69$	Keep-alive & break configuration. See table **a** below.	1 to 4

Default value : $_{h}00$

a. Keep-alive configuration bits

Offset	Length	Content
0	1	Keep-alive & break configuration bits. See table **b** below
1	0 to 3	Value of constant frame for keep-alive.

b. *Keep-alive & Break options*

Bit	Value	Meaning
7 – 6	00 01 10 11	Break switches disabled Front break switch enabled Back break switch enabled Both break switches enabled
5	0 1	If keep-alive disabled : Send alarm frame[6] If keep-alive enabled : Stop keep-alive frame Send alarm state frames[7]
4	 0 1	***Keep-alive frame :*** Send an empty frame Send a constant frame – Value of constant frame starts at offset 1
3 – 0	 0000 0001 to 1111	***Keep-alive interval :*** Keep-alive disabled Delay between 2 keep-alive frames. Minimum = $_{h}$1 (1s) to maximum = $_{h}$F (15s)

Information about value of alarm state :

- Alarm state = $_{h}$00, both break switches are trigged.
- Alarm state = $_{h}$01, front break switch is broken.
- Alarm state = $_{h}$02, back break switch is broken.
- Alarm state = $_{h}$03, both break switches are broken.

2.2.9. 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.

[6] 1 byte is to send when there is a break :

<Start frame marker> <Alarm state> <Stop frame marker>

[7] Add 1 byte to keep-alive frames :

- If keep-alive empty frame (bit 4 = 0) :

  <Start frame marker> <Alarm state> <Stop frame marker>

- If keep-alive constant frame (bit 4 = 1) :

  <Start frame marker> <Value of constant> <Alarm state> <Stop frame marker>

2.3. CARD PROCESSING TEMPLATES

Each Card Processing Template is configured through a set of 16 tags, from $_{h}$t0 to $_{h}$tF where 't' is the template group ($_{h}1 \leq t \leq {}_{h}4$).

2.3.1. *Card lookup list*

Name	Tag	Description	Size
LKL	$_{h}$t0	Card lookup list of the template. See table **a** below.	1

a. *Available values for LKL*

Value	Card(s) accepted by the template	Processing template	§
$_{h}$01	ISO/IEC 14443 type A (layer 3)	**ID only**	2.4
$_{h}$02	ISO/IEC 14443 type B (layer 3)		
$_{h}$03	ISO/IEC 14443 A&B (layer 3)		
$_{h}$04	ISO/IEC 15693		
$_{h}$07	ISO/IEC 14443 A&B and ISO/IEC 15693		
$_{h}$08	NXP ICODE1		
$_{h}$0C	NXP ICODE1 and ISO/IEC 15693		
$_{h}$0F	All of the above		
$_{h}$11	ISO/IEC 14443 type A (layer 4 / T=CL)	**7816-4**	2.8
$_{h}$12	ISO/IEC 14443 type B (layer 4 / T=CL)		
$_{h}$13	ISO/IEC 14443 A&B (layer 4 / T=CL)		
$_{h}$22	ST MicroElectronics SR family	**ID only**	2.4
$_{h}$23	ASK CTS256B and CTS512B		
$_{h}$24	Inside Contactless PicoTAG[8]		
$_{h}$61	NXP Mifare Classic 1k & 4k	**Mifare**	2.5
$_{h}$62	NXP Mifare UltraLight	**Mifare UltraLight**	2.6
$_{h}$71	NXP Desfire 4k	**Desfire**	2.7
$_{h}$72	Calypso (Innovatron protocol)	**ID only or 7816-4**	2.9

Other values are *RFU*

The LKL tag is mandatory to enable a template group. If not found, the template group is empty.

[8] Also HID iClass

2.3.2. *Summary of other tags in templates*

Depending of the card lookup list (LKL tag), a specific list of tags controls the behaviour of the Processing Template.

The table below summarize this.

<table>
<tr><th>Tag</th><th>ID only</th><th>Mifare UL</th><th>Mifare</th><th>Desfire</th><th>7816-4</th><th>Calypso</th></tr>
<tr><td>$_{h}$t1</td><td colspan="6">Output format</td></tr>
<tr><td>$_{h}$t2</td><td colspan="6">Output prefix</td></tr>
<tr><td>$_{h}$t3</td><td>Offset</td><td colspan="5">Location of data</td></tr>
<tr><td>$_{h}$t4</td><td></td><td></td><td></td><td colspan="2">T=CL options</td><td>C. options</td></tr>
<tr><td>$_{h}$t5</td><td></td><td></td><td colspan="2">Auth. method & key</td><td colspan="2">1st APDU</td></tr>
<tr><td>$_{h}$t6</td><td></td><td></td><td colspan="2">Sign. method & key</td><td colspan="2">2nd APDU</td></tr>
<tr><td>$_{h}$t7</td><td></td><td></td><td colspan="2"></td><td colspan="2">3rd APDU</td></tr>
</table>

Grey items are *RFU* and must be kept empty.

2.3.3. *Important notice regarding template-ordering*

Be careful that the 4 templates are processed one after the other. The loop is ended after the first successful match.

If a card matches two (or more) templates, it will be handled only by the first one.

Suppose you want to accept both a specific kind of 14443-B T=CL cards, with advanced file reading, and another kind of wired-logic 14443-B cards, where only the ID is significant. You must put the T=CL template *before* the ID template, otherwise the T=CL part will be skipped.

2.4. ID-ONLY PROCESSING TEMPLATE

2.4.1. *Lookup list*

Name	Tag	Description	Size
LKL.IDO	$_h$t0	**ID-only lookup list :** $_h$01 ≤ value ≤ $_h$0F for ISO-compliant cards, $_h$21 ≤ value ≤ $_h$2F for non-ISO cards. See **2.3.1a** for details.	1

2.4.2. *Output format*

Name	Tag	Description	Size
TOF.IDO	$_h$t1	ID-only output format. See table **a** below.	1

a. *Output format bits*

Bit	Value	Meaning
7 – 6		**Byte swapping**
	00	Do not swap ID bytes (ID is transmitted “as is”)
	01	*RFU*
	10	Swap bytes for single-size (4 bytes) ISO 14443-A UIDs [9] only ; IDs of any other card is transmitted “as is”
	11	Swap ID bytes for all kind of cards
5		**Padding**
	0	Left-padding with $_h$0
	1	Right-padding with $_h$F
4		**ISO 14443-B specific**
	0	Use ISO 14443-B PUPI (4 bytes) as ID
	1	Use complete ISO 14443-B ATQ (11 bytes) as ID
3 – 0		**Output length**
	0000	Decimal, 4 bytes seen as 10 digits (i.e. 32 → 40 bits expansion)
	0001	Fixed length, 4 bytes [10]
	0010	Fixed length, 8 bytes [11]
	0011	Fixed length, 5 bytes
	0100	Fixed length, 12 bytes [12]
	0101	Fixed length, 7 bytes [13]
	0110	Fixed length, 11 bytes [14]
	0111	*RFU*
	1000	Fixed length, 16 bytes
	1001	*RFU*
	1010	*RFU*
	1011	*RFU*
	1100	Decimal, 5 bytes seen as 12 digits (i.e. 40 → 56 bits expansion)
	1101	Decimal, 5 bytes seen as 13 digits (i.e. 40 → 64 bits expansion)
	1110	Decimal, variable length (maximum 13 digits)
	1111	Variable length (depends on actual size of ID)

Default value : $_b$10000010

(8 bytes fixed length, left padding, swap bytes for short ISO 14443-A UIDs only)

2.4.3. *Output prefix*

Name	Tag	Description	Size

[9] This is the default format in NXP’s Mifare Classic related literature.

[10] ISO 14443-A single-size UID, ISO 14443-B PUPI, serial number for ASK CTS256B and CTS512B.

[11] ISO 15693 ID, serial number for NXP ICODE1, Inside Contactless PicoTag, ST MicroElectronics SR family…

[12] ISO 14443-A triple-size UID.

[13] ISO 14443-A double-size UID.

[14] ISO 14443-B complete ATQB.

PFX.IDO	$_h$t2	ID-only output prefix.	Var.

Default value : absent *(no prefix)*

If a non-null ASCII value is specified (either a single character or a string), it will be transmitted before the data (therefore the actual length will be longer than the specified length).

2.4.4. Offset of data

Name	Tag	Description	Size
LOC.IDO	$_h$t3	Offset in the ID.	1

Default value : $_b$00000000 ($_d$0)

When TOF.IDO specifies a fixed length output, using LOC.IDO makes it possible to select some bytes in the ID, and not only the first ones. This is principally useful when working with non-ISO cards, see 2.4.5 for details.

2.4.5. Non-ISO cards

A few manufacturers offers non standard cards, most of them based on ISO 14443-B bit-level specification, but with a proprietary frame format (protocol) and a proprietary command set.

As those cards don't answer to ISO 14443 standard detection commands, a specific template must be activated to discover them.

a. ST MicroElectronics SR family

When LKL.IDO=$_h$22, the reader performs the lookup sequence for cards in the ST MicroElectronics SR family (SR176, SRX, SRIX).

A 8-byte serial number is returned by the card. Use TOF.IDO and LOC.IDO if you need to truncate it.

b. ASK CTS256B and CTS512B

When LKL.IDO=$_{h}$23, the reader performs the lookup sequence for cards in the ASK CTS-B family (CTS256B, CTS512B).

A 8-byte identifier is built as follow :

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Bytes 4 to 7**
Manufacturing code	Product code	Embedded code	Application code	4-byte serial number

- CTS256B's product code is between $_{h}$50 and $_{h}$5F,
- CTS512B's product code is between $_{h}$60 and $_{h}$6F,
- See ASK's documentation for explanations regarding other bytes.

Define LOC.IDO=$_{h}$04 (and TOF.IDO=$_{h}$01) if you need the serial number only (without card type and other data).

c. Inside Contactless PicoTAG[15]

When LKL.IDO=$_{h}$24, the reader performs the lookup sequence for cards in the Inside Contactless PicoTag family (PicoTag 16KS).

A 8-byte serial number is returned by the card. Use TOF.IDO and LOC.IDO if you need to truncate it.

[15] Also HID iClass

2.5. MIFARE CLASSIC PROCESSING TEMPLATE

Mifare "Classic" refers to NXP's Mifare 1k and Mifare 4k wired-logic contactless cards.

Mifare 1k is divided into 64 16-byte blocks.

Mifare 4k is divided into 256 16-byte blocks.

Both cards have a 4-byte serial number, located at the beginning of block 0. As those cards are ISO/IEC 14443-3 compliant, you can read the serial number through the generic ID-Only template, instead of using this dedicated template.

2.5.1. *Lookup list*

Name	Tag	Description	Size
LKL.MIF	$_h$t0	Mifare classic lookup list, value = $_h$61. See **2.3.1a** for details.	1

2.5.2. *Output format*

Name	Tag	Description	Size
TOF.MIF	$_h$t1	Mifare output format. See table **a** below.	1

a. Output format bits

Bit	Value	Meaning
7	0	Do not swap bytes
	1	Swap bytes
6	0	RAW data
	1	ASCII encoded data [16]
5	0	Left-padding with $_h$0 (RAW) or <SPACE> (ASCII)
	1	Right-padding with $_h$F (RAW) or <SPACE> (ASCII)
4		*RFU*
3 – 0		**Output length** Format depends on bit 6 (RAW or ASCII). See table **b** below for RAW data (bit 6 = 0) See table **c** below for ASCII data (bit 6 = 1)

Default value : $_b$00000010

[16] If data read from the memory card is "31 32 33 43 34 35" (hexadecimal notation), output will be "123C45". Make sure that only valid digits (values from 31 to 39 and 41 to 46 or 61 to 66) are encoded in every card, otherwise actual reader output will be undefined.

b. *Output length when bit 6 = 0*

Bit	Value	Meaning
3 – 0	0000	Decimal, 4 bytes seen as 10 digits (i.e. 32 → 40 bits expansion)
	0001	Fixed length, 4 bytes (32 bits)
	0010	Fixed length, 8 bytes (64 bits)
	0011	Fixed length, 5 bytes (40 bits)
	0100	Fixed length, 12 bytes (96 bits)
	0101	Fixed length, 7 bytes (56 bits)
	0110	Fixed length, 11 bytes (88 bits)
	0111	*RFU*
	1000	Fixed length, 16 bytes (128 bits)
	1001	*RFU*
	1010	*RFU*
	1011	*RFU*
	1100	Decimal, 5 bytes seen as 12 digits (i.e. 40 → 56 bits expansion)
	1101	Decimal, 5 bytes seen as 13 digits (i.e. 40 → 64 bits expansion)
	1110	Decimal, variable length (maximum 13 digits)
	1111	Variable length (using ${}_{h}0$ and ${}_{h}F$ as end of string markers)

c. *Output length when bit 6 = 1*

Bit	Value	Meaning
3 – 0	0000	Max output length = ${}_{d}16$
	0001 to 1111	Max output length from ${}_{d}1$ to ${}_{d}15$

2.5.3. *Output prefix*

Name	Tag	Description	Size
PFX.MIF	${}_{h}$t2	Mifare output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.5.4. Location of data

Depending on the size, the LOC.MIF tag can either be

- A block number (= address of data in Mifare card) when size = 1,
- An Application Identifier (AID) when size = 2.

a. Fixed block number

Name	Tag	Description	Size
LOC.MIF	$_{h}$t3	Block number to be read.	1

Default value : $_{b}$00000100 ($_{d}$4)

When a Mifare card is found, the reader tries to read the block specified in LOC.MIF (16 bytes), and then truncates the data according to the length specified in TOF.MIF.

The block number shall be

- Between 0 and 63 for Mifare 1k cards,
- Between 0 and 255 for Mifare 4k cards.

Note that data must start on a block boundary.

Mifare sector trailers (security blocks) numbered 3, 7, … can be read, but their content is masked (to protect the keys). Using such a block as access control identifier is definitely not a good idea.

b. AID in MAD

Name	Tag	Description	Size
LOC.MIF	$_{h}$t3	AID to be selected and read.	2

When a Mifare card is found, reader reads the MAD (blocks 1 and 2 of sector 0)[17] and tries to find the specified AID. The location of the AID in the MAD is the pointer onto the actual block to be read.

Note that data must be located at the beginning of the first block marked with the specified AID.

Please refer to NXP application notes for detailed explanations of the MAD.

[17] Sector 0 must be freely readable either with base key A ("A0 A1 A2 A3 A4 A5"), with transport key ("FF FF FF FF FF FF") or with the application key specified in AUT.MIF .

2.5.5. Authentication key

Depending on the size, the AUT.MIF tag can either be

- A pointer to a key located in RC's secure EEPROM when size = 1.
- The Mifare key itself, when size = 7,
- A master key and its diversification options, when size = 9 or 17

When the AUT.MIF tag is absent, all EEPROM keys are tried out in sequence (this can take a long time…).

Name	Tag	Description	Size
AUT.MIF	$_h$t5	Mifare authentication key.	See below

Default value : absent

a. Size = 1 : pointer to a key in RC's secure EEPROM

- Values $_h$00 to $_h$0F refer to type A keys $_d$0 to $_d$15, respectively,
- Values $_h$10 to $_h$1F refer to type B keys $_d$0 to $_d$15, respectively.

b. Size = 7 : specified Mifare key

Offset	Length	Content
0	1	Key options. See table **c** below.
1	6	Mifare key value.

c. Key options bits, when size = 7

Bit	Value	Meaning
7	0 1	Key is an A key Key is a B key
6 – 0		*RFU*

d. Size = 17 : master key diversification using HMAC-MD5

Offset	Length	Content
0	1	Key options. See table **e** below.
1	16	Master key value.

e. Key options bits, when size = 17

Bit	Value	Meaning
7	0 1	Diversified key is an A key Diversified key is a B key
6	0 1	Diversification with card UID and address fixed to $_h$00 Diversification with card UID and address = sector number
5 – 4	10	Diversify the key using HMAC-MD5 algorithm *(see chapter 9)*
3 – 0		*RFU*

f. Size = 15 or 23 : master key diversification using RC171 algorithm

Offset	Length	Content
0	1	Key options. See table **g** below.
1	6	Mifare master key.
7	8 or 16	DES or 3-DES diversification key.

g. Key options bits, when size = 15 or 23

Bit	Value	Meaning
7	0 1	Diversified key is an A key Diversified key is a B key
6	0 1	Diversification with card UID and address fixed to $_h$00 Diversification with card UID and address = sector number
5 – 4	01	Diversify the key using RC171 algorithm *(see chapter 10)*
3 – 0		*RFU*

2.6. MIFARE ULTRALIGHT PROCESSING TEMPLATE

NXP's Mifare UltraLight is a low-cost wired-logic contactless cards. It is divided into 16 4-byte pages. This template reads 4 pages (i.e. exactly 16 bytes) at once.

This card has a 7-byte serial number, located on blocks 0 and 1. As the card is ISO/IEC 14443-3 compliant, you can read the serial number through the generic ID-Only template, instead of using this dedicated template.

2.6.1. Lookup list

Name	Tag	Description	Size
LKL.MFU	$_{h}$t0	Mifare UltraLight lookup list, value = $_{h}$62. See **2.3.1a** for details.	1

2.6.2. Output format

Name	Tag	Description	Size
TOF. MFU	$_{h}$t1	Mifare UltraLight output format.	1

Same as Mifare Classic output format (2.5.2).

2.6.3. Output prefix

Name	Tag	Description	Size
PFX.MFU	$_{h}$t2	Mifare UltraLight output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.6.4. Location of data

Name	Tag	Description	Size
LOC.MFU	$_{h}$t3	Number of the first page to be read.	1

Default value : $_{b}$00000000 ($_{d}$0)

Remember that this template always reads 4 pages (16 bytes) starting at LOC.MFU.

2.7. DESFIRE CARD PROCESSING TEMPLATE

2.7.1. Lookup list

Name	Tag	Description	Size
LKL.DFR	$_h$t0	Desfire lookup list, value = $_h$71. See **2.3.1a** for details.	1

2.7.2. Output format

Name	Tag	Description	Size
TOF.DFR	$_h$t1	Desfire output format.	1

Same as Mifare Classic output format (2.5.2).

2.7.3. Output prefix

Name	Tag	Description	Size
PFX.DFR	$_h$t2	Desfire output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.7.4. Location of data

Name	Tag	Description	Size
LOC.DFR	$_h$t3	Location of data in Desfire card. See table **a** below.	8

a. Data location bytes

Offset	Length	Content
0	3	Application IDentifier (AID).
3	1	File IDentifier (FID). File must be a "standard data" file.
4	3	Offset of data in file.
7	1	Length of data to be read[18] (1 to 64).

Default value : unspecified.

Values are MSB first.

[18] Data will be truncated to the length specified in TOF.DFR .

2.7.5. T=CL options

Name	Tag	Description	Size
OPT.DFR	$_h$t4	Desfire T=CL options.	1

Same as 7816-4 T=CL options (2.8.5).

2.7.6. Authentication key

Name	Tag	Description	Size
AUT.DFR	$_h$t5	Desfire authentication key. See table **a** below.	9 or 17

Default value : absent

(No authentication is performed, plain read operation is used to fetch the data)

a. Authentication key bytes

Offset	Length	Content
0	1	Desfire key index and options. See table **b** below.
1	8 or 16	Key value (8 bytes for a DES key, 16 bytes for a 3-DES key).

b. Key index and options

Bit	Value	Meaning
7 – 6		**Communication mode for reading**
	00	Plain
	01	MACed with session key
	10	*RFU*
	11	Enciphered with session key
5 – 4		**Key diversification algorithm**
	00	Use the key "as is"
	01	Diversify the key using Desfire SAM algorithm *(see chapter 10)*
	10	Diversify the key using HMAC-MD5 algorithm *(see chapter 9)*
	11	*RFU*
3 – 0		**Index of key in Desfire application**
	0000 to 1110	Index of the key to be used for authentication
	1111	*RFU*

2.8. 7816-4 CARD PROCESSING TEMPLATE

2.8.1. Lookup list

Name	Tag	Description	Size
LKL.TCL	$_h$t0	7816-4 lookup list, $_h$11 ≤ value ≤ $_h$13. See **2.3.1a** for details.	1

2.8.2. Output format

Name	Tag	Description	Size
TOF.TCL	$_h$t1	T=CL output format.	1

Same as Mifare Classic output format (2.5.2).

2.8.3. Output prefix

Name	Tag	Description	Size
PFX.TCL	$_h$t2	T=CL output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.8.4. Location of data

Name	Tag	Description	Size
LOC.TCL	$_h$t3	Offset of data in answer to APDU 3[19] (0 to 127).	1

Default value : 0.

2.8.5. T=CL options

Name	Tag	Description	Size
OPT.TCL	$_h$t4	T=CL (ISO/IEC 14443 layer 4) options. See table **a** below.	1

[19] Data will be truncated according to the length specified in TOF.TCL .

a. T=CL option bits

Bit	Value	Meaning
7 – 6		**Card to reader baudrate**
	00	No PPS, DSI = 106kbit/s
	01	Perform PPS, DSI = 212kbit/s if card allows it
	10	Perform PPS, DSI = 424kbit/s if card allows it
	11	Perform PPS, DSI = 848kbit/s if card allows it
5 – 4		**Reader to card baudrate**
	00	No PPS, DRI = 106kbit/s
	01	Perform PPS, DRI = 212kbit/s if card allows it
	10	Perform PPS, DRI = 424kbit/s if card allows it
	11	Perform PPS, DRI = 848kbit/s if card allows it
3 – 0		**Card identifier (CID)**
	0000	Empty CID = $_d$0
	0001 to 1110	CID from $_d$1 to $_d$14
	1111	CID is disabled

This tag exists only if T=CL card is selected in LST.

Default value : $_b$00001111

2.8.6. T=CL APDU 1

Typically this is a Select Application (or Select Applet) command.

May be absent if T=CL APDU 3 is sufficient to fetch the data.

Name	Tag	Description	Size
AU1.TCL	$_h$t5	TCL APDU 1.	Var.

✋ Card's Status Word is checked by the reader. A SW between $_h$9000 and $_h$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between $_h$6100 and $_h$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.8.7. T=CL APDU 2

Typically this is a Select File command.

May be absent if T=CL APDU 3 is sufficient to fetch the data.

Name	Tag	Description	Size
AU2.TCL	${}_{h}$t6	TCL APDU 2.	Var.

Card's Status Word is checked by the reader. A SW between ${}_{h}$9000 and ${}_{h}$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between ${}_{h}$6100 and ${}_{h}$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.8.8. T=CL APDU 3

APDU used to actually retrieve the data (typically this is a Read Binary command). Data have to be found in answer at offset specified in LOC.TCL.

Name	Tag	Description	Size
AU3.TCL	${}_{h}$t7	TCL APDU 3.	Var.

Card's Status Word is checked by the reader. A SW between ${}_{h}$9000 and ${}_{h}$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between ${}_{h}$6100 and ${}_{h}$6FFF) is considered as an error, and the reader will ignore the card.

Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.9. CALYPSO CARD PROCESSING TEMPLATE

This part deals with old Calypso cards, to be accessed only through the legacy Innovatron radio protocol.

New Calypso cards now support ISO/IEC 14443-B, and therefore can be accessed either through ID-Only or ISO/IEC 7816-4 templates.

> ✋ Working with Calypso cards is subject to a specific licence fee. This function is therefore disabled for out-of-factory readers.
>
> Please contact us to have the Calypso functionality enabled in your readers.

Depending on the specified options, this Calypso card processing template can retrieve :

- A 4-byte serial number (ID-Only template)
- Arbitrary data to be read in Calypso files (7816-4 template)

2.9.1. *Lookup list*

Name	Tag	Description	Size
LKL.CYO	$_h$t0	Calypso/Innovatron lookup list, value = $_h$72. See **2.3.1a** for details.	1

2.9.2. *Output format*

Name	Tag	Description	Size
TOF.CYO	$_h$t1	Calypso/Innovatron output format.	1

Same as Mifare Classic output format (2.5.2).

2.9.3. *Output prefix*

Name	Tag	Description	Size
PFX.CYO	$_h$t2	Calypso/Innovatron output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.9.4. Location of data

Name	Tag	Description	Size
LOC.CYO	$_h$t3	Offset of data in answer to APDU 3[20] (0 to 64).	1

Default value : 0.

2.9.5. Calypso APDU 1

Typically this is a Select DF command.

Name	Tag	Description	Size
AU1.CYO	$_h$t5	Calypso/Innovatron APDU 1.	Var.

> ✋ Card's Status Word is checked by the reader. A SW between $_h$9000 and $_h$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between $_h$6100 and $_h$6FFF) is considered as an error, and the reader will ignore the card.
>
> Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

2.9.6. Calypso APDU 2

Typically this is a Select EF command.

Name	Tag	Description	Size
AU2.CYO	$_h$t6	Calypso/Innovatron APDU 2.	Var.

> ✋ Card's Status Word is checked by the reader. A SW between $_h$9000 and $_h$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between $_h$6100 and $_h$6FFF) is considered as an error, and the reader will ignore the card.
>
> Reader's internal buffer is limited to 128 bytes. If card's answer is longer, the answer will be discarded and the reader will ignore the card.

[20] Data will be truncated according to the length specified in TOF.CYO .

2.9.7. Calypso APDU 3

Typically this is a Read Binary command.

Name	Tag	Description	Size
AU3.CYO	${}_{h}$t7	Calypso/Innovatron APDU 3	Var.

✋ Card’s Status Word is checked by the reader. A SW between ${}_{h}$9000 and ${}_{h}$9FFF is considered valid. Any other value for SW (and in particular error values as defined by ISO 7816-4 between ${}_{h}$6100 and ${}_{h}$6FFF) is considered as an error, and the reader will ignore the card.

Reader’s internal buffer is limited to 128 bytes. If card’s answer is longer, the answer will be discarded and the reader will ignore the card.

2.10. SUMMARY OF CONFIGURATION TAGS

Name	Tag	Content
	$_h$10 $_h$11 … $_h$1F	**Card Processing Template #1** (out of factory : versatile ID-only reader)
	$_h$20 $_h$21 … $_h$2F	**Card Processing Template #2** (out of factory : empty)
	$_h$30 $_h$31 … $_h$3F	**Card Processing Template #3** (out of factory : empty)
	$_h$40 $_h$41 … $_h$4F	**Card Processing Template #4** (out of factory : empty)
	$_h$50 $_h$51 … $_h$5F	**Reserved for Master Cards** (see chapter 7)
OPT	$_h$60	General configuration
ODL	$_h$61	Output delay
RDL	$_h$62	Repeat delay
CLD	$_h$63	LEDs control configuration
CBZ	$_h$64	Buzzer control configuration
WGD	$_h$65	Output configuration when reader works in Wiegand mode
DTC	$_h$66	Output configuration when reader works in Dataclock mode
SER	$_h$67	Output configuration when reader works in RS-232/485/USB mode
SHD	$_h$68	Output configuration when reader works in RS-485 mode
KAL	$_h$69	Configuration of keep-alive frame
PIN	$_h$6F	Console access PIN code

3. CONFIGURING IWM-X

There are two ways to configure IWM-X :

- Using a Master Card, formatted with **cfgfilecreator.exe** software. See chapter 8 for details,
- Manually, by entering configuration values in reader's console (serial line access), as shown below.

In both cases, three of the four jumpers enable or prevent LEDs and buzzer operation.

The first jumper enables the serial line access for console operation.

Whatever hardware is used, default factory settings for IWM-X firmware are :

- RS-485 mode, 38400bps,
- Reads any kind of ID, 8 byte fixed length output.

Always configure IWM-X properly before installation as there're little chances that default configuration matches your requirements.

3.1. CONNECTING IWM-X TO A COMPUTER

3.1.1. IWM-X-10

IWM-X-01 features both IrDA interface and a 8-pin PC-link connector (serial interface at TTL level). Pro-Active offers 2 USB interface to connect the reader to a computer running Microsoft Windows.

a. IrDA

Use SpringCard **INT-USB-ITL** as a USB interface through IrDA. Using a computer's built-in IrDA interface is unsupported.

b. 8-pin PC-link connector

Use either SpringCard **INT-USB-TTL** or **INT-USB-ITL** as a USB interface using reader's PC-Link Connector.

c. Driver

Install software ref. **SDD100** "USB Driver for Pro-Active's FTDI-based devices" to see the INT-USB-ITL or INT-USB-TTL interface as a virtual serial port (VCP).

3.1.2. IWM-X-20

Connect the reader directly to the computer using either the serial line (RS-232) or the USB interface (type B connector).

When using the USB interface, install software ref. **SDD100** "USB Driver for Pro-Active's FTDI-based devices" to see the reader as a virtual serial port (VCP).

3.1.3. Common information

Use HyperTerminal or any compliant terminal emulator to get connected onto the reader through the serial port. Default communication settings are :

- 8 data bits, 1 stop, no parity, no flow control ;
- Baudrate = 38400bps[21].

3.1.4. Testing connection

- Move the corresponding switch to "trace-enabled mode" position,
- Power-up (or reset) the reader,
- Reader sends its identification string :

```
SpringCard IWM-X 1.25
```

3.2. RETRIEVING IWM-X INFORMATION

3.2.1. Firmware version

Enter "`ver`" to read IWM-X firmware version.

3.2.2. Firmware configuration

Enter "`sho`" to read IWM-X configuration.

[21] Baudrate may be altered by configuration, but default baudrate (38400bps) is always restored when the "trace-enabled mode" switch is set.

3.3. ENABLING CONFIGURATION COMMANDS

IWM-X configuration may be protected by a pin-code (if PIN configuration tag is empty, no pin-code is needed.

If defined to $_{h}$FFFF, configuration commands are permanently disabled).

Enter "`pinNNNN`" to allow configuration commands, where NNNN is the actual pin-code (for instance, "`pin1234`")[22].

3.4. ACCESSING IWM-X CONFIGURATION

3.4.1. *Reading configuration tags*

Enter "`cfg`" to list all configuration tags.

Enter "`cfgXX`" to read value configuration tag XX (hexadecimal address).

Note that configuration tags $_{h}$55, $_{h}$56 and $_{h}$6F (keys used by Master Cards and pin-code) are masked when read back.

3.4.2. *Writing configuration tags*

Enter "`cfgXX=YYYY`" to update configuration tag XX (hexadecimal address) with value YYYY (hexadecimal value).

Enter "`cfgXX=!!`" to delete configuration tag XX (hexadecimal address).

3.4.3. *Writing Mifare keys in MfRC's secure EEPROM*

Enter "`keya0=XXXXXXXXXXXX`" to update key A at index 0, "`keya1=...`" to update key A at index 1, and so on until "`keyaf=...`".

Enter "`keyb0=XXXXXXXXXXXX`" to update key B at index 0, "`keyb1=...`" to update key B at index 1, and so on until "`keybf=...`".

Note that keys stored in RC can't be read back.

3.4.4. *Reading MfRC's 4-byte EEPROM*

RC's chipset includes a 4-byte EEPROM to store a configuration value.

Enter "`cfgRC`" to read this 4-byte value.

[22] For security reasons, configuration commands are enabled only for 3 minutes. After 3 minutes of inactivity, you'll have to enter the pin-code again.

3.4.5. Writing MfRC's 4-byte EEPROM

RC's chipset includes a 4-byte EEPROM to store a configuration value.

Enter "`cfgRC=XXXXXXXX`" to write this 4-byte value.

Content of RC's 4-byte EEPROM is currently not used by IWM-X firmware.

Please keep this value to 00000000 as it may be used in future versions.

3.5. APPLYING NEW CONFIGURATION

New configuration is applied only after reset.

Cycle power or enter "`rst`" to reset the reader.

Information in this document is subject to change without notice. Reproduction without written permission of PRO ACTIVE is forbidden. PRO ACTIVE and the PRO ACTIVE logo are registered trademarks of PRO ACTIVE SAS. All other trademarks are property of their respective owners.

3.6. REVERTING TO DEFAULT

Sometimes it is necessary to put reader back in "out-of-factory" configuration (for instance when reader goes from one site to another). This is done easily by erasing all tags from reader's memory.

Enter "`cfg!!=!!`" to delete all configuration tags.

☠	There's no confirmation prompt nor any kind of "are you sure ?" popup window. Erasing everything is immediate and unrecoverable.

✋	Erasing all the configuration tags is not really enough to put the reader(s) back in out-of-factory configuration, since Mifare keys stored in RC's secure EEPROM are not erased. Read paragraph 3.5.3 to see how the keys may be overwritten.

4. SERIAL MODE APPLICATION NOTE

4.1. THE RS-485 INTERFACE

IWM-X-10 only

a. Pins

Pins 3 and 4 are RS-485 A and RS-485 B, respectively.

b. Communication parameters

Default baudrate is 38400bps. See 2.5.1 if you need to change this value.

Other parameters are :

- 8 data bits, 1 stop bit
- No parity, no flow control.

Those parameters can't be altered.

4.2. THE RS-232 AND USB INTERFACES

IWM-X-20 only

Default baudrate is 38400bps. See 2.5.1 if you need to change this value.

Other parameters are :

- 8 data bits, 1 stop bit
- No parity, no flow control.

Those parameters can't be altered.

4.3. SERIAL OUTPUT

4.3.1. Frame markers

Serial frame markers are configured by bits 7-5 of SER .

a. When addressing is disabled

Consider data '01 23 45 67',

- If bits 7-5 = $_b$000, frame is "01234567".

- If bits 7-5 = $_b$001, frame is "01234567<CR><LF>" where <CR> the ASCII carriage return ($_h$0D), and <LF> the ASCII line feed ($_h$0A).
- If bits 7-5 = $_b$010, frame is "<BEL>01234567<CR><LF>" where <BEL> is the ASCII bell (or ring) character ($_h$07), <CR> the ASCII carriage return ($_h$0D), and <LF> the ASCII line feed ($_h$0A).
- If bits 7-5 = $_b$011, frame is "<TAB>01234567<CR><LF>" where <TAB> is the ASCII horizontal tab character ($_h$09), <CR> the ASCII carriage return ($_h$0D), and <LF> the ASCII line feed ($_h$0A).
- If bits 7-5 = $_b$100, frame is "<STX>01234567<ETX>" where <STX> is the ASCII "start of text" character ($_h$02), and <ETX> the ASCII "end of text" ($_h$03).
- If bits 7-5 = $_b$101, frame is "<STX>01234567<ETX><CR><LF>".
- If bits 7-5 = $_b$110, frame is "<BEL><STX>01234567<ETX><CR><LF>".
- If bits 7-5 = $_b$111, frame is "<TAB><STX>01234567<ETX><CR><LF>".

b. *When addressing is enabled*

Consider data '01 23 45 67' and address 'a' ($_h$1 ≤ a ≤ $_h$E),

- If bits 7-5 = $_b$000, frame is "a>01234567".
- If bits 7-5 = $_b$001, frame is "a>01234567<CR><LF>".
- If bits 7-5 = $_b$010, frame is "<BEL>a>01234567<CR><LF>".
- If bits 7-5 = $_b$011, frame is "<TAB>a>01234567<CR><LF>".
- If bits 7-5 = $_b$100, frame is "<SOH>a><STX>01234567<ETX>" where <SOH> is the ASCII "start of header" character ($_h$01).
- If bits 7-5 = $_b$101, frame is "<SOH>a><STX>01234567<ETX><CR><LF>".
- If bits 7-5 = $_b$110, frame is "<BEL><SOH>a><STX>01234567<ETX><CR><LF>".
- If bits 7-5 = $_b$111, frame is "<TAB><SOH>a><STX>01234567<ETX><CR><LF>".

4.4. SERIAL INPUT

IWM-X accepts short commands from the host, to drive LEDs and buzzer output mostly.

IWM-X doesn't echo received data (unless "console mode" jumper is ON).

If received command has been understood by IWM-X, it replies with <ACK> before executing the requested action. Otherwise, it replies with <NACK>.

4.4.1. When addressing is disabled

Command transmission format is <command> <CR> <LF>.

4.4.2. When addressing is enabled

Command transmission format is <address> **<** <command> <CR> <LF>, where <address> must be the address of the device.

4.4.3. List of commands

Command	***Action***
A0	Reader goes inactive (tag polling is halted)
A1	Reader goes active
R0	Switch red LED off
R1	Switch red LED on
R2	Red LED blinks slowly
R3	Red LED blinks quickly
G0	Switch green LED off
G1	Switch green LED on
G2	Green LED blinks slowly
G3	Green LED blinks quickly
Z0	Stop buzzer
Z1	Start buzzer
Z2	Short buzzer sound
Z3	Long buzzer sound
M*argz*	Same as sending A*a* + R*r* + G*g* + Z*z*
M*rg*	Same as sending R*r* + G*g*
M*arg*	Same as sending A*a* + R*r* + G*g*
RST	Reset the reader
VER	Retrieve reader’s version
SHO	Retrieve reader’s settings

5. WIEGAND APPLICATION NOTE

IWM-X-10 only

5.1. THE WIEGAND INTERFACE

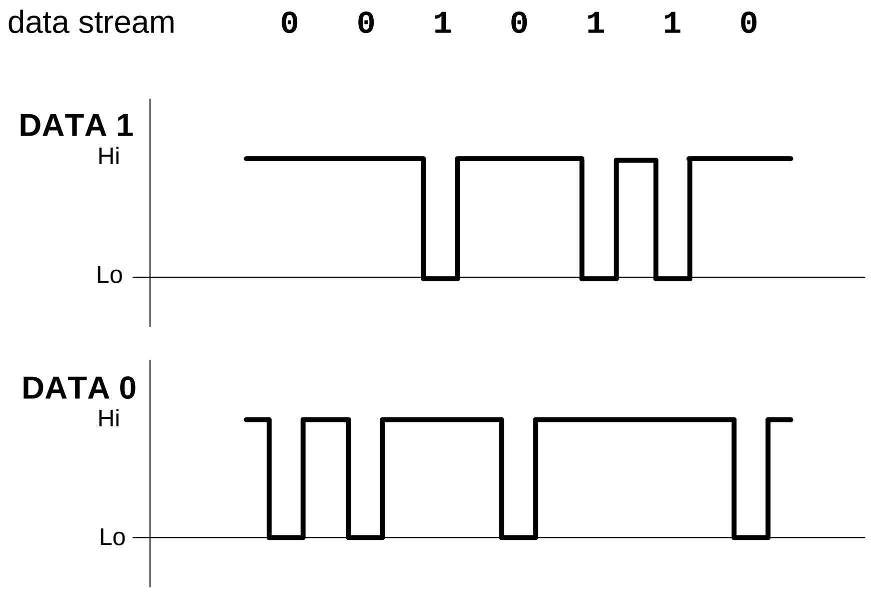
a. Pins

Pins 5 and 6 are Wiegand DATA0 and DATA1 outputs, respectively.

- Both pins are at high level when idle,
- A low pulse on DATA0 denotes a bit 0 output,
- A low pulse on DATA1 denotes a bit 1 output.

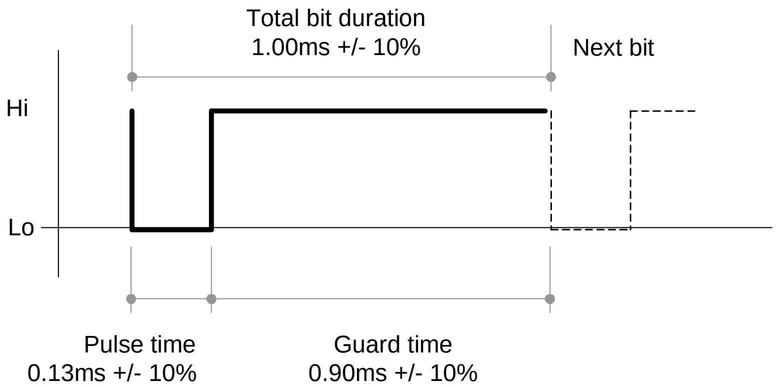
In normal operation, DATA0 and DATA1 are never at low level simultaneously.

b. Signals



c. Timings

Those are the default timings ; they can be altered by writing into WGD configuration tag :



It is better that the controller triggers on the falling edge of the signal or on the low level, and not on the rising edge.

5.1.2. *Electrical levels*

DATA0 and DATA1 are open collector outputs. Pull-up resistors (R_{ctrl}) <u>must</u> be supplied by the controller (to V_{ctrl} level).

Maximum values are :

- V_{ctrl} = 15V
- I_{max} = 10mA

Typical values are :

- V_{ctrl} = 5V R_{ctrl} = 4,7kΩ (I_{max} = 1,0mA)
- V_{ctrl} = 12V R_{ctrl} = 10kΩ (I_{max} = 1,2mA)

5.2. FRAME FORMAT

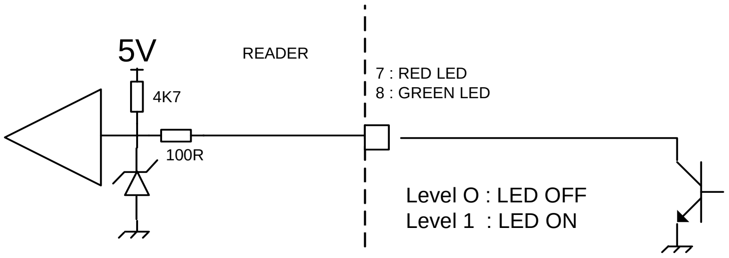
Wiegand output format is fully driven by the templates.

5.3. LED INTERFACE

Pins 7 and 8 are respectively red and green LEDs inputs.

	Meaning	*Level to GND*
Input level high	LED is ON	3.3V to 5.5V
Input level low	LED if OFF	0.0V to 1.7V

The reader has an internal pull-up resistor to 5V.



✋ Choose proper configuration in CLD to allow an external control of the LEDs.

6. DATACLOCK APPLICATION NOTE

IWM-X-10 only

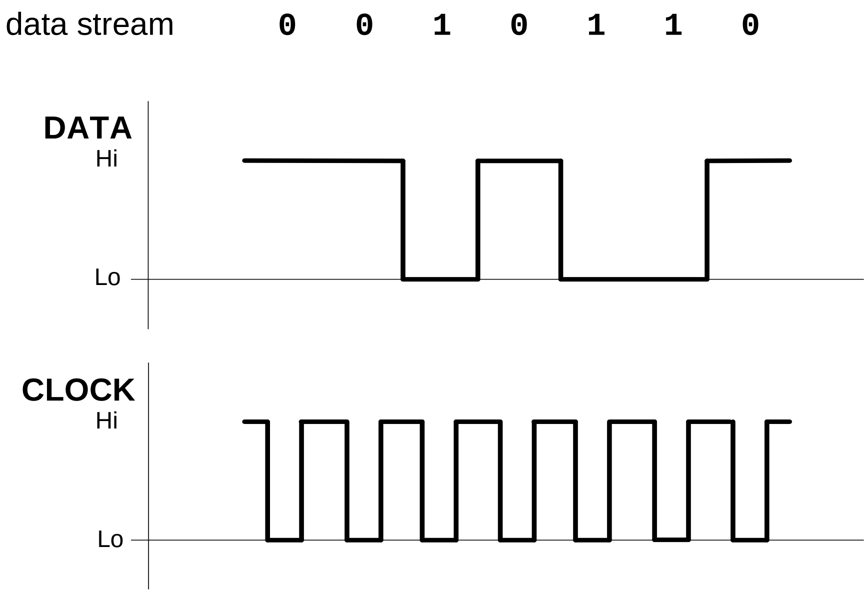
6.1. THE DATACLOCK INTERFACE

a. Pins

Pins 5 and 6 are Wiegand DATA0 and DATA1 outputs, respectively.

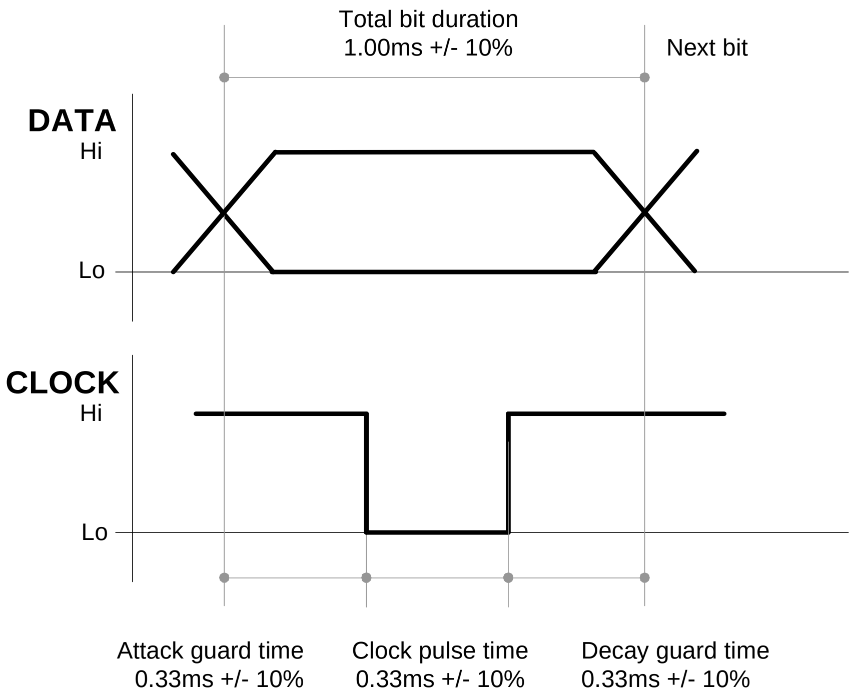
- Both pins are at high level when idle,
- The CLOCK line is active low,
- The DATA line is inverting (low level means 1, high level means 0).

b. Signals



c. Timings

Default clock period is 1ms (approx.) with a duty cycle 1/3 (330µs inactive, 330µs active low, 330µs inactive). Those timings can be altered by writing into DTC configuration tag.



It is better that the controller triggers on the falling edge of the clock or on the low level, and not on the rising edge.

6.1.2. Electrical levels

DATA and CLOCK are open collector outputs. Pull-up resistors (R_{ctrl}) must be supplied by the controller (to V_{ctrl} level).

Maximum values are :

- V_{ctrl} = 15V
- I_{max} = 10mA

Typical values are :

- V_{ctrl} = 5V R_{ctrl} = 4,7kΩ (I_{max} = 1,0mA)
- V_{ctrl} = 12V R_{ctrl} = 10kΩ (I_{max} = 1,2mA)

6.1.3. Digit format

Dataclock only transmit decimal data. Each digit is transmitted as 5 bits :

- 4 digit bits, least significant bit first,
- 1 parity bit.

Data are BCD-encoded, i.e. only decimal values from 0 to 9 are valid for data digits. Values above 10 (hexadecimal values from A to F) are reserved.

Dataclock digit format

Value	*Bit pattern*

Value	*Bit pattern*	*Reserved for*

0	0 0 0 0 1
1	1 0 0 0 0
2	0 1 0 0 0
3	1 1 0 0 1
4	0 0 1 0 0
5	1 0 1 0 1
6	0 1 1 0 1
7	1 1 1 0 0
8	0 0 0 1 0
9	1 0 0 1 1

A (10)	0 1 0 1 1	
B (11)	1 1 0 1 0	Start sentinel
C (12)	0 0 1 1 1	
D (13)	1 0 1 1 0	Separator
E (14)	0 1 1 1 0	
F (15)	1 1 1 1 1	Stop sentinel

6.2. ISO2 / MAGSTRIPE FRAMES

6.2.1. *Frame content*

When the ISO2 / Magstripe format is selected (bit 7 = 0 in DTC), only decimal digits (0 to 9) are allowed. This is OK when data read from the card is actually decimal numbers.

In case data is not composed of numbers but arbitrary binary values, a translation must be applied before actual transmission. This translation is defined by bits 3-2 of DTC.

Consider the data '00 7A 12 6C 59 F4 04' in hexadecimal notation (this is the serial number of a Mifare Ultralight card). Digits 'A' and 'F' are not allowed in the frame.

a. *Discard non-decimal*

- If bits 3-2 = $_b$00, frame will be '00712659404'.

b. *Replace by separators*

- If bits 3-2 = $_b$01, frame will be '007-126-59-404' where '-' is the dataclock separator character (digit $_h$D).

c. *Translation method 1*

- If bits 3-2 = $_b$10, frame will be '0000071001020601250915040004'. Note that each data digit (hexadecimal $_h$0 to $_h$F) has been replaced by two decimal digits ($_d$00 to $_d$15). Frame length is twice as big as data length.

d. *Translation method 2*

- If bits 3-2 = $_b$11, frame will be '007-0126-259-5404'. Note that valid decimal digits have been transmitted "as is", where digits from $_h$A to $_h$F ($_d$10 to $_d15$) have been replaced by the '-' separator followed by the divided-by-10 reminder.

6.2.2. Frame prefix and postfix

ISO2/Magstripe frames are transmitted according to following protocol :

1. Left edge : bit 0 is transmitted 16 times,
2. Start sentinel (hexadecimal digit B, i.e. bit pattern "1 1 0 1 0"),
3. Actual frame content as specified in 2.2.5,
4. Stop sentinel (hexadecimal digit F, i.e. bit pattern "1 1 1 1 1"),
5. LRC of frame (XOR computed over parts 1, 2 and 3),
6. Right edge : bit 0 is transmitted 16 times.

6.3. RAW FRAMES

6.3.1. Frame content

When the RAW format is selected (bit 7 = 1 in DTC), data are sent "as is", any digit from $_d0$ ($_h0$) to $_d15$ ($_hF$) being allowed.

6.3.2. Frame prefix and postfix

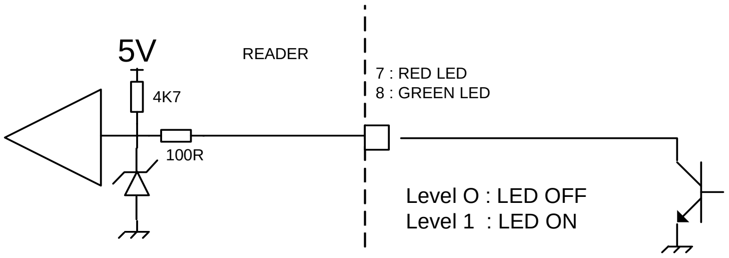
RAW frames are transmitted without prefix and postfix.

6.4. LED INTERFACE

Pins 7 and 8 are red and green LEDs inputs, respectively.

	Meaning	*Level to GND*
Input level high	LED is OFF	3.3V to 5.5V
Input level low	LED if ON	0.0V to 1.7V

The reader has an internal pull-up resistor to 5V.



✋ Choose proper configuration in CLD to allow an external control of the LEDs.

7. SPECIFICATION OF MASTER CARDS

> 👍 This chapter is provided as a mean for security experts to evaluate IWM-X Master Card architecture.
>
> Customers do not need to implement this part themselves, since **cfgfilecreator.exe** software is a convenient tool to create Master Cards. See chapter 8 for details.

7.1. BUILDING A MASTER CARD

- The Master Card must be a Desfire 4k,
- The reader tries to fetch configuration data from Desfire cards according to the Master Card template specified in next paragraph. Data are protected by an authentication key that may be changed on a per-customer or per-site basis (i.e. Master Cards belonging to customer X will not work on customer Y's readers),
- Before storing new settings in its non-volatile memory, the reader checks that data comes with a valid digital signature. The signing key can't be changed, and is only known by Pro-Active's software. This ensure that only data that has been pre-validated by a genuine software can be loaded in reader's non-volatile memory.

7.2. TEMPLATE FOR MASTER CARDS

7.2.1. *Location of data*

Name	Tag	Description	Size
LOC.MAS	$_h$53	Location of data in master cards. See table **a** below.	5

a. Data location bytes

Offset	Length	Content	Specified value
0	3	Application IDentifier (AID).	$_h$504143
3	1	File IDentifier (FID) for configuration data.	$_h$01
4	1	File IDentifier (FID) for digital signature.	$_h$02

7.2.2. Authentication key

Out-of-factory key used for authentication of Master Cards is confidential.

Only Pro-Active genuine software –such as **cfgfilecreator.exe**– is able to create Master Cards with the default authentication key.

To secure their installation, customers should replace this key as soon as they receive the readers, as explained in 8.4 .

This is the same structure as AUT.DFR .

Name	Tag	Description	Size
AUT.MAS	${}_{h}$55	Authentication key. See table **a** below.	17

a. Authentication key bytes

Offset	Length	Content
0	1	Authentication key index and options. See table **b** below.
1	16	***Authentication key for Master Cards*** (this is 3-DES key).

b. Authentication key index and options

Bit	Value	Meaning
7 – 6		**Communication mode in read operation**
	00	Plain
	01	MACed with session key
	10	*RFU*
	11	Enciphered with session key
5 – 4		**Key diversification algorithm**
	00	Use the key “as is”
	01	Diversify the key using Desfire SAM algorithm *(see chapter 10)*
	10	Diversify the key using HMAC-MD5 algorithm *(see chapter 9)*
	11	*RFU*
3 – 0		**Index of key in Desfire application**
	0000 to 1110	Index of the key to be used for authentication
	1111	*RFU*

Specified value : ${}_{h}$E0 *(key 0, HMAC-MD5 diversification, ciphered reading)*

7.2.3. Signing key

Name	Tag	Description	Size
SGN.MAS	${}_{h}56$	Signing key. See table **a** below.	17

☠ Key used for digital signature of master cards is confidential.

Only Pro-Active genuine software –such as **cfgfilecreator.exe**– is able to sign the Master Cards[23].

Customers shall not try to change this parameter, unless advised to by Pro-Active.

a. Signing key bytes

Offset	Length	Content
0	1	Index and options. See table **b** below.
1	16	***Key data*** (this is 128-bits key).

b. Signing key index and options

Bit	Value	Meaning
7 – 6	00	*Those bits are RFU and must be 00*
5 – 4	 00 01 10 11	**Key diversification algorithm** Use the key "as is" Diversify the key using Desfire SAM algorithm *(see chapter 10)* Diversify the key using HMAC-MD5 algorithm *(see chapter 9)* *RFU*
3 – 0	0000	*Those bits are RFU and must be 00*

Specified value : ${}_{h}20$ *(HMAC-MD5 diversification)*

[23] This choice has been done to ensure that data inside the Master Card have been pre-validated according to reader specifications, and have not been corrupted afterwards.

7.3. DATA STRUCTURE

7.3.1. Size of file

File holding configuration data and Mifare keys (offset 3 in LOC.MAS) must be exactly 512-byte long. In case used size is shorter than 512 bytes, file must be padded with $_h$00.

7.3.2. Configuration data

The configuration data block uses the T,L,V (tag, length, value) encoding scheme.

- Tag is 1 byte-wide,
- Len is 1 byte-wide,
- Value is 0 to 24 byte-wide.

Items found in T,L,V blocks will overwrite data with the same tag already present in reader's non-volatile memory.

Set Len = 0 to delete an existing tag from the non-volatile memory, without replacing it.

Last T,L,V of the configuration data block must be the digital signature of the whole block, according to the algorithm specified in 7.4.

7.3.3. Mifare keys to be loaded into RC's secure EEPROM

Keys to be loaded into RC's secure EEPROM use the T,L,V scheme, as follow :

- Tag (1 byte) = $_h$80 + key index as specified in 2.6.4.a,
- Len (1 byte) = $_h$06,
- Value is the Mifare key (6 bytes exactly).

7.4. DIGITAL SIGNATURE

7.4.1. Size of file

File holding the signature (offset 4 in LOC.MAS) must be exactly 16-byte long.

7.4.2. Algorithm

This is the signature algorithm when default parameters in SGN.KEY as used :

- Let *Content* be the 512-byte configuration block as written in the card[24],
- Let *SignKey* be the 16-byte key,
- Diversify *SignKey* from card's UID, using HMAC-MD5 diversification algorithm[25] to get *DivKey*,
- Compute *Sign* = HMAC-MD5 (*Block*) using *DivKey* [26].

As specified in 7.2.3, value of *SignKey* is confidential. Customers shall not try to change the key, nor the signature algorithm.

[24] This is the configuration data plus the Mifare keys to be loaded into RC's secure EEPROM. Total size is up to 512 bytes (as required by 7.3.1). Note that signature is computed over the whole file, including its padding, whatever the used length is.

[25] See 8.3.1

[26] See 8.2

8. USING IWMK632 SOFTWARE TO CREATE MASTER CARDS

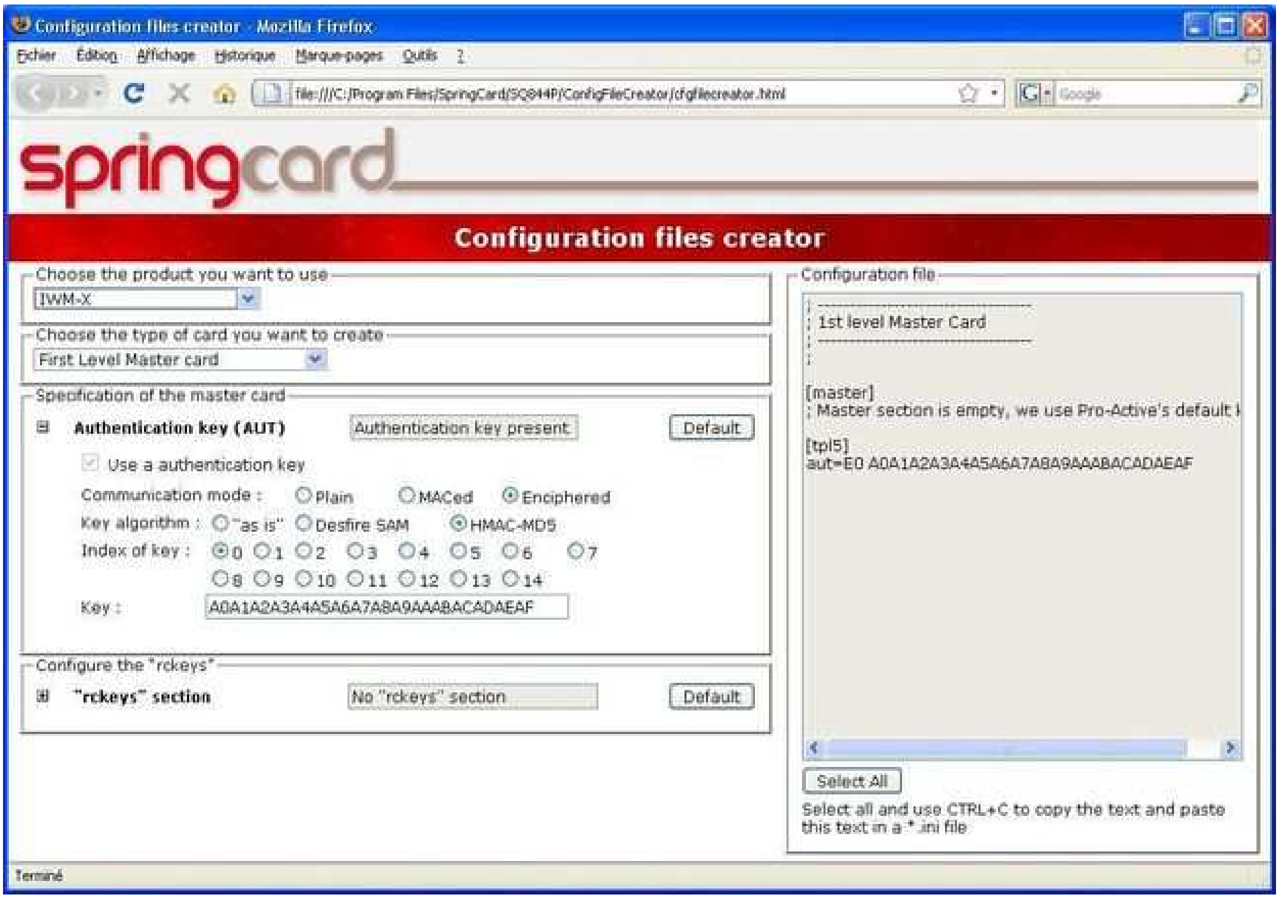
8.1. OVERVIEW

Master Cards are NXP Desfire 4k. You may buy them from Pro-Active or any other NXP reseller.

cfgfilecreator.exe is a command line software (running on Microsoft Windows) to create Master Cards. **cfgfilecreator.exe** needs a SpringCard Prox'N'Roll PC/SC (or legacy) contactless coupler to program the cards.

> Enter **cfgfilecreator.exe -h** to read the complete list of command line switches and options, and the complete list of sections and variables for configuration files.
>
> **cfgfilecreator.exe** software comes with various sample configuration files that show typical configurations of IWM-X.

cfgfilecreator.html is a standalone web page that helps creating configuration files for **cfgfilecreator.exe** .


Configuration files creator - Mozilla Firefox
Fichier Edition Affichage Historique Marque-pages Outils ?
file:///C:/Program Files/SpringCard/SQ844P/ConfigFileCreator/cfgfilecreator.html
Google
springcard
Configuration files creator
Default
Default
Select All
Terminé

Choose the product you want to use: IWM-X

Choose the type of card you want to create: First Level Master card

Specification of the master card

Authentication key (AUT): Authentication key present

☑ Use a authentication key

Communication mode : ○ Plain ○ MACed ● Enciphered

Key algorithm : ○ "as is" ○ Desfire SAM ● HMAC-MD5

Index of key : ● 0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 ○ 11 ○ 12 ○ 13 ○ 14

Key : A0A1A2A3A4A5A6A7A8A9AAABACADAEAF

Configure the "rckeys"

"rckeys" section: No "rckeys" section

Configuration file

```
; -----------------------------------
; 1st level Master Card
; -----------------------------------
;

[master]
; Master section is empty, we use Pro-Active's default k

[tpl5]
aut=E0 A0A1A2A3A4A5A6A7A8A9AAABACADAEAF
```

Select all and use CTRL+C to copy the text and paste this text in a *.ini file

8.2. CONFIGURATION FILES

cfgfilecreator.exe uses a configuration file to retrieve configuration data to be written into the Master Card.

Configuration files are written like standard Windows "INI" files. They can be created using Notepad or any other text editor, or using **cfgfilecreator.html** .

Each line of each section uses the format "name=value" where "name" is either the name or the tag of the configuration variable (e.g. either "opt" or "60"), and "value" its value in hexadecimal.

8.2.1. The "general" section

This section maps to tags $_{h}$60 to $_{h}$6F. Default content is :

```
[general]
opt=05      ; value for OPT
odl=02      ; value for ODL
rdl=0A      ; value for RDF
cld=0F      ; value for CLD
cbz=13      ; value for CBZ
wgd=0A      ; value for WGD
dtc=0A      ; value for DTC
ser=C5      ; value for SER
shd=00      ; value for SHD
pin=0000    ; value for PIN
```

8.2.2. The "rckeys" section

This section holds the Mifare access keys to be written in RC's secure EEPROM.

Type A keys are named "a0" to "a15", and type B keys "b0" to "b15".

Here's an example of content :

```
[rckeys]
a0=A0A1A2A3A4A5 ; Mifare type A base key (for MAD)
a1=FFFFFFFFFFFF ; NXP transport key
a2=000000000000 ; other transport key
a3=CCCCCCCCCCCC ; unused
(...)
a15=CCCCCCCCCCCC ; unused
b0=B0B1B2B3B4B5 ; Mifare type B base key (for MAD)
b1=FFFFFFFFFFFF ; NXP transport key
b2=000000000000 ; other transport key
b3=CCCCCCCCCCCC ; unused
(...)
b15=CCCCCCCCCCCC ; unused
```

This section (and each line in it) is optional. Only keys listed in this section will be written, other keys will be left unchanged.

8.2.3. Sections for Card Processing Templates

IWM-X may run from 1 to 4 card accepting templates. Each template is configured by sections "tpl1", "tpl2", "tpl3" and "tpl4" respectively.

Mandatory and optional content for each section depends on the card lookup list (LKL field) of the section itself.

a. ID-Only example

This sample section configures template 4 to read any kind of ID. Output format is : 8-byte fixed length, prefixed by the string "ID=" :

```
[tpl4]
lkl=0F                   ; wants any kind of ID
tof=82                   ; 8-byte output, swap 14443 A short IDs
pfx=49443D               ; prefix = "ID="
```

b. Desfire example

This sample section configures template 1 to read 8 bytes of data from a Desfire card. Output format is : 8-byte fixed length, no prefix :

```
[tpl1]
lkl=71                  ; wants Desfire cards
tof=02                  ; 8-byte output
pfx=                    ; no prefix
loc=123456 01 000100 08 ; 8 bytes of data to be read in application
                        ; 0x123456, field 0x01, at offset 0x000100
aut=00 A0A1A2A3A4A5A7   ; authentication with key 0, plain comm.
                        ; mode, no diversification. Key is a single
                        ; DES key (8 bytes)
```

8.2.4. Master Cards related sections

a. Specifying a new configuration for future Master Cards

The "tpl5" section allows to update the card processing template reserved to Master Cards. See paragraph 8.4.1 for details.

```
[tpl5]
aut=E0 xx...xx        ; 16-byte authentication key
```

> ✋ This 16-byte authentication key in the "tpl5" section is the one that will be written in the reader(s) by the Master Card.
>
> It is not the key that will be used to create the Master Card itself.

b. Specifying configuration to be used by current Master Card

The "master" section defines how the Master Card shall be created. See paragraph 8.4.2 for details.

```
[master]
aut=E0 xx...xx        ; 16-byte authentication key
```

> ✋ This 16-byte authentication key in the "master" section is the one that will be used to create the Master Card.
>
> It has no impact on the key written in the reader(s).

8.3. OPERATION INSTRUCTIONS

- Open **Configuration files creator** (**cfgfilecreator.html**)
(on Windows : Start Menu → All Programs → SpringCard → Configuration Tools),
- Create your configuration file and save it in the directory where **cfgfilecreator.exe** is installed, for instance with the name *siteconf.ini* (on Windows : C:\Program Files\SpringCard\SQ844P),
- Open **Configuration tools directory**
(on Windows : Start Menu → All Programs → SpringCard → Configuration Tools),
- Plug and power-on your Prox'N'Roll PC/SC (or legacy),
- Put a virgin Desfire card on the Prox'N'Roll PC/SC (or legacy),
- Enter **cfgfilecreator.exe –c *siteconf.ini***,
- Wait until Master Card is written.

☠ If the Desfire card is not virgin, the **software will try to format it** (i.e. erase the whole file structure with all the data) **without prior notification**.

Be sure to put on the reader only a virgin card, or an old Master Card to be overwritten.

You've been warned…

8.4. CHANGING AUTHENTICATION KEY FOR MASTER CARDS

🔒	All IWM-X ship with the same out-of-factory authentication key. To secure their site, customers should replace the default key by their own key before installing the readers.

Pro-Active recommends to make (and keep) at least two distinct Master Cards for each customer or site :

- ***1st level Master Card*** alters only the authentication key (replace default key by site specific key).
  - All readers bought for this site shall be configured using this *1st level Master Card* as soon as they are received.
- ***2nd level Master Card*** actually configures the reader (card processing templates, output mode and format, and so on).
  - It uses the site specific key for authentication, but doesn't update the key that is already inside the reader.
  - The *2nd level Master Card* shall be used during installation and whenever you wish to change reader configuration.

Note that many *2nd level Master Cards* can be created (one for each kind of output settings, one for each people in charge of installation…) whereas only one *1st level Master Card* should be created and be kept in a secure place[27].

☠	Be sure to remember the new authentication key you put in a reader. If you forget the authentication key, and forget the pin-code (or define pin-code to $_h$FFFF), it will be impossible to change reader configuration again ! **You've been warned…**

[27] That's because *1st level Master Card* has got the authentication key written in it, and anybody may retrieve it using **cfgfilecreator** software, where the authentication key is only used to secure *2nd level Master Cards* and is not written in them.

8.4.1. *Creating a first level Master Card*

- Create a configuration file (say, "*master.ini*") with only those 4 lines :

```
[master]
; Master section is empty, we use Pro-Active's default keys

[tpl5]
aut=E0 xx...xx
```

where *xx…xx* is the site specific 16-byte authentication key[28],

- Put a virgin card on the Prox'N'Roll, label it "1st level Master Card",
- Enter **cfgfilecreator.exe –c *master.ini*** ,
- Use this Master Card to write the new authentication key in the reader(s).

8.4.2. *Creating a second level Master Card*

- Create a complete configuration file as seen in § 8.3 .
- Terminate the file with those 4 lines :

```
[master]
aut=E0 xx...xx

[tpl5]
; Template 5 section is empty, we keep current keys in the reader
```

where *xx…xx* is the site specific 16-byte authentication key[28],

- Put a virgin card on the Prox'N'Roll, label it "2nd level Master Card",
- Enter **cfgfilecreator.exe –c *siteconf.ini*** ,
- Use this Master Card to write complete configuration in the reader(s).

[28] This is key 0 inside Master Card application ; the key will be diversified using HMAC-MD5 algorithm, so the "E0" header is mandatory.

8.5. REVERTING TO DEFAULT

Sometimes it is necessary to put reader back in "out-of-factory" configuration (for instance when reader goes from one site to another). This is done easily by erasing all tags from reader's memory.

- Create a configuration file (say, "*factory.ini*") with only those 3 lines :

```
[master]
aut=E0 xx...xx
clear=1
```

  where *xx…xx* is the site specific 16-byte authentication key

- Put a virgin card on the Prox'N'Roll, label it "Erase all Master Card",
- Enter **cfgfilecreator.exe –c *factory.ini***
- Use this Master Card to put the reader(s) back in out-of-factory configuration.

> ✋ Erasing all the configuration tags is not really sufficient to put the reader(s) back in out-of-factory configuration, since Mifare keys stored in RC's secure EEPROM are not erased.
>
> Just add an "rckeys" section (as specified in 8.2.2), with dummy keys, to overwrite those keys.

9. HMAC SIGNATURE AND KEY DIVERSIFICATION

9.1. HMAC-MD5

9.1.1. *Abstracts*

A message authentication code, or MAC, is a short piece of information used to authenticate a message. A MAC algorithm accepts as input a secret key and a message, and outputs a MAC that protects both message's integrity and authenticity.

An HMAC (or keyed-hash message authentication code) is a type of MAC function were a cryptographic hash function is used to compute the output.

9.1.2. *Algorithm*

$$\mathrm{HMAC}_K(m) = h\Big((K \oplus \mathrm{opad}) \| h\big((K \oplus \mathrm{ipad}) \| m\big)\Big),$$

Where *h* is the hash function, *K* is the secret key padded with extra zeros up to 64 bytes, *m* is the message to be authenticated. *opad* is the value $_h$5C repeated 64 times, and ipad the value $_h$36 repeated 64 times.

HMAC-MD5 is a particular HMAC function where *h* is the MD5 standard function, as defined by RSA laboratories. Size of HMAC is 16 bytes exactly.

9.2. USING HMAC-MD5 FOR SIGNATURE

HMAC protects both message's integrity and authenticity, so it can be considered as a digital signature[29].

IWM implementation allows only 16-byte keys. The key can be used "as is" or be the result of a diversification from a master key.

9.3. USING HMAC-MD5 FOR KEY DIVERSIFICATION

In this particular mode, we name *K* the "master key" and we compute the HMAC over card's identifier to establish a "diversified key" *Ku*.

[29] Literature often reserve the name "digital signature" to public key schemes, where verifier doesn't need to know signer's private key to verify the signature. HMAC is a scheme where signer and verifier must share the same secret key.

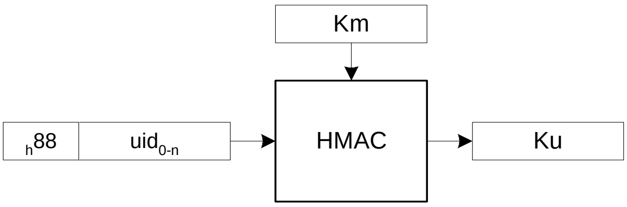
9.3.1. DES & Triple-DES key diversification algorithm

The algorithm takes as inputs :

- A 16-byte master key (Km)
- The card serial number (uid)[30]

It provides as output :

- The 16-byte diversified key specific to this card (Ku).



The diversified key can now be used either for Desfire authentication, or for HMAC-MD5 signature.

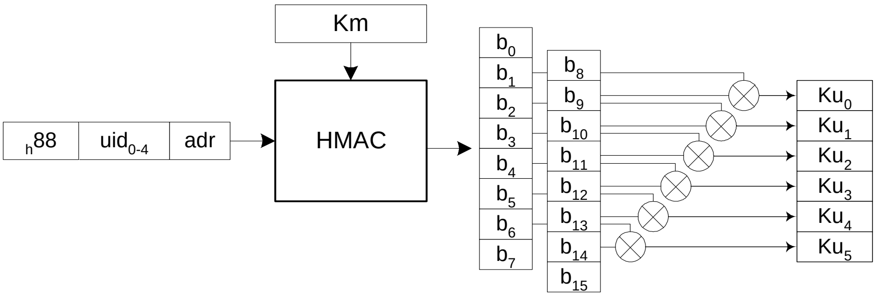
9.3.2. Mifare key diversification algorithm

The algorithm takes as inputs :

- A 16-byte master key (Km)
- The 4-byte card serial number (uid)
- The 1-byte block address (adr)

It provides as output :

- The 6-byte Mifare key specific to the couple card + address (Ku).



See last two paragraphs of chapter 10, for details regarding how the *adr* parameter shall be understood.

[30] The UID is 7-byte long for a Desfire card, 4-byte long for a Mifare card. The same diversification algorithm is usable whatever the length is.

10. DESFIRE SAM & RC171 KEY DIVERSIFICATION

10.1. DES AND 3-DES KEY DIVERSIFICATION

The key diversification algorithm described here is the one provided by Desfire SAM. Please refer to the corresponding datasheet for details.

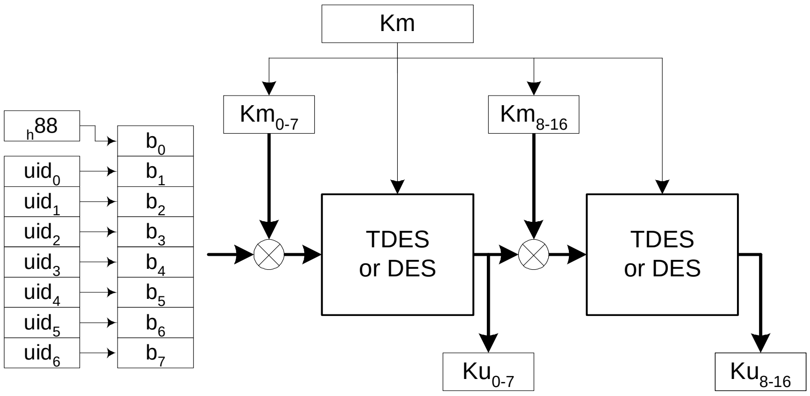
The algorithm takes as inputs :

- A 16-byte Triple-DES master key (Km)[31]
- The 7-byte card serial number (uid)

It provides as output :

- The 16-byte diversified key specific to this card (Ku).

Here's the flowchart :



The diversified key now be used for Desfire authentication.

[31] If both halves are equals, the key maps to a single DES key

10.2. MIFARE KEY DIVERSIFICATION

The Mifare diversification algorithm described here is provided both by Desfire SAM and by RC171 secure coprocessor. Please refer to the corresponding datasheets for details.

10.2.1. Basis

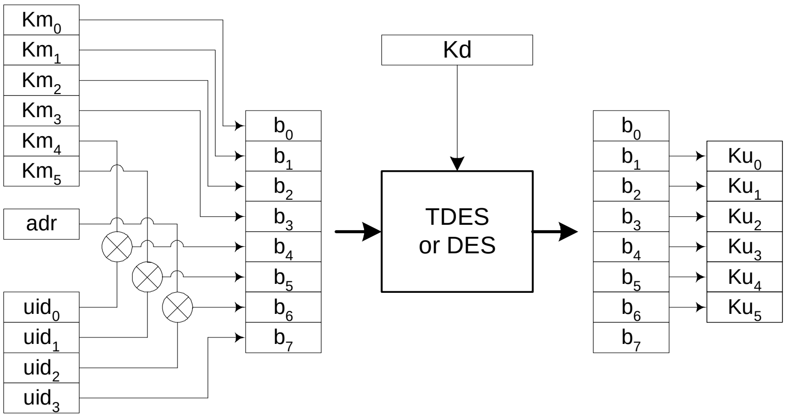
The algorithm takes as inputs :

- A 6-byte master key (Km)
- A 16-byte Triple-DES diversification key (Kd)[32]
- The 1-byte block address (adr)
- The 4-byte card serial number (uid)

It provides as output :

- The 6-byte Mifare key specific to the couple card + address (Ku).

Here's the flowchart :



10.2.2. Diversification based on UID only

If this option is selected, the *adr* input parameter is fixed to $_h00$ whatever block to be read is.

[32] If both halves are equals, the key maps to a single DES key

10.2.3. Diversification based on UID and address

If this option is selected, the *adr* input parameter is the Mifare sector number.

Here's an example with a Mifare 1k card :

- Data is located on block 29,
- Block 29 belongs to sector 7 (29 / 4),
- The diversification algorithm will be fed with adr = 7.

Here's an example with a Mifare 4k card :

- Data is located on block 231,
- Block 231 belongs to sector 38 (32 + (231-128) / 16),
- The diversification algorithm will be fed with adr = 38.

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 does not 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

Published by **Pro-Active SAS**, 13, voie La Cardon 91120 Palaiseau – France

R.C.S. EVRY B 429 665 482 - APE 26127

For more information, please contact us at info@pro-active.fr .