IWM-K632 Wall-mount contactless reader

Reference manual

PMAA061 revision AC
28/01/2008

TABLE OF CONTENT

1. INTRODUCTION

This document provides detailed technical information for use of the Pro-Active wall-mount contactless proximity card reader IWM-K632.

1.1. AUDIENCE

This reference manual assumes that the reader has expert knowledge of electronics. It is designed for use by system integrators.

1.2. PRODUCT BRIEF

IWM-K632 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. This reader is primarily dedicated to corporate access control, where an high level of security or versatility is needed, but can also be used in cash or vending machines.

IWM-K632 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. 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 (**IWM-K632-WD** or **IWM-K632-SU**), the serial input/output can either be RS-232, RS-485, or USB (USB to serial bridge).

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-K632 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-K632 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-K632 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-K632 reader.

2.2. GLOBAL SETTINGS

The following tables enumerates 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[4] serial half-duplex (RS-485) reader[4] Wiegand reader[5] Dataclock reader[5]

Default value : ${}_{b}10000101$

(power saving mode, Master Cards enabled only at power up, 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

[4] Actual RS-232, RS-422, RS-TTL or USB compliance depends on hardware.

[5] USER output pin is supposed to drive a RS-485 buffer. Actual RS-485 compliance depends on hardware.

2.2.2. Delays and repeat options

Name	Tag	Description	Min	Max
ODL	h61	Min. delay between 2 consecutive outputs (tenth of seconds)	0	100
RDL	h62	Min. delay between 2 consecutive identical outputs (tenth of seconds). 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	LED sequences last 3 seconds LED sequences last 10 seconds
6	0 1	No detection of host controller Both LEDs flash until host controller is detected[6]
5	0 1	When idle, red LED blinks slowly ("heart beat" sequence) When idle, red LED is off
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 discovered in the field
1 – 0	00 01 10 11	LED control by hardware lines, other settings are ignored[7] LED control by serial commands, other settings are ignored[6 & 8] RFU LED control by internal software and serial commands[6 & 8]

Default value : b00001111

[6] Valid for serial modes only

[7] Jumpers 2 & 3 must be set to OFF

[8] Jumpers 2 & 3 must be set to ON

b. Buzzer control bits[9]

Bit	Value	Meaning
7	0 1	Buzzer short pulse last 0,2 sec. Buzzer short pulse last 0,5 sec.
6	0 1	Buzzer long pulse last 0,7 sec. Buzzer long pulse last 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 discovered 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	$_h$65	Wiegand configuration bits. See table **a** below.	1

a. Wiegand configuration bits

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

Default value : $_b$00001010

See chapter 5.1 for details on Wiegand timings.

[9] Set jumper 4 to ON to allow buzzer control. If jumper 4 is OFF, buzzer is totally disabled.

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[10] Raw output (bits 3-2 are ignored)[11]
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.

[10] 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.

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

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

Name	Tag	Description	Size
SER	ₕ67	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	0	No BEL / CR/LF frame markers
	1	Use BEL and CR/LF as frame markers
5 – 3		RFU
2 – 0	000	Baudrate = 1200bps
	001	Baudrate = 2400bps
	010	Baudrate = 4800bps
	011	Baudrate = 9600bps
	100	Baudrate = 19200bps
	101	Baudrate = 38400bps
	110	RFU
	111	Baudrate = 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 7-6 drives 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. 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 values $_{h}$FFFF that permanently disables the console feature.

2.3. CARD PROCESSING TEMPLATES

Each Card Processing Template is configured through a bunch 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}$41	NXP Mifare UltraLight	Memory	2.5
${}_{h}$42	STMicroElectronics SR176		
${}_{h}$43	ASK CTS256 and CTS512		
${}_{h}$61	NXP Mifare Classic 1k & 4k	Mifare	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.

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>Memory</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></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. Note on template order

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.

For instance, suppose you want to accept both a specific kind of 14443-B T=CL cards, with advanced file reading, and another kind of dumb 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 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	0	Do not revert short type A UIDs
	1	Revert short type A UIDs (LSB first instead of MSB first) [12]
6	0	Do not revert long type A UIDs
	1	Revert long type A UIDs (LSB first instead of MSB first)
5	0	Left-padding with $_h$0
	1	Right-padding with $_h$F
4		RFU
3		RFU
2 – 0		*Output length*
	000	Decimal, 4 bytes seen as 10 digits (i.e. 32 → 40 bits expansion)
	001	Fixed length, 4 bytes [13]
	011	Fixed length, 7 bytes [14]
	010	Fixed length, 8 bytes
	101	Fixed length, 11 bytes [15]
	110	Fixed length, 12 bytes [16]
	100	Fixed length, 16 bytes
	111	Variable length (depends on actual size of ID)

Default value : $_b$10000010

(8 bytes fixed length, left padding, revert short type A UIDs)

[12] This is the default format in NXP & Mifare literature.

[13] Type A single size UID, type B PUPI.

[14] Type A double size UID.

[15] Type B complete ATQB.

[16] Type A triple size UID.

2.4.3. Output prefix

Name	Tag	Description	Size
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).

2.5. MEMORY CARD PROCESSING TEMPLATE

2.5.1. *Lookup list*

Name	Tag	Description	Size
LKL.MEM	$_{h}$t0	Memory lookup list, $_{h}41 \leq$ value $\leq {}_{h}43$ See **2.3.1a** for details.	1

2.5.2. *Output format*

Name	Tag	Description	Size
TOF.MEM	$_{h}$t1	Memory 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 [17]
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

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)
	0100	Fixed length, 12 bytes (96 bits)
	1000	Fixed length, 16 bytes (128 bits)
	0011	Fixed length, 5 bytes (40 bits)
	0101	Fixed length, 7 bytes (56 bits)
	1111	Variable length (using $_{h}$0 and $_{h}$F as end of string markers)
		Other values are RFU

[17] 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 cards, over wise actual reader output will be undefined.

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.MEM	$_{h}$t2	Memory output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.5.4. Location of data

Name	Tag	Description	Size
LOC.MEM	$_{h}$t3	Location of data in memory card.	1

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

When a supported memory card is found, reader tries to read data starting at the address specified in LOC.MEM, and up to length bytes (fixed length specified in TOF.MEM).

The actual interpretation of LOC.MEM depends on the card type, see application notes in the next paragraphs.

2.5.5. Mifare UltraLight application note

Mifare UltraLight is structured as 16 x 4-byte pages (numbered 0 to 15). LOC.MEM specifies the number of the first page to be read (note that the data must be aligned on a page boundary). 4 pages (i.e. exactly 16 bytes) are read by this template before formatting.

a. Reading serial number

The 7-byte serial number is located at the beginning of address space (4 bytes of page 0 + 3 bytes of page 1).

As Mifare UltraLight is ISO/IEC 14443-3 compliant, you can also process it through the appropriate ID-Only template.

b. Reading other data

Make sure that 4 x LOC.MEM + length specified in TOF.MEM doesn’t exceed the actual capacity of the card, i.e. 64 bytes.

2.5.6. ST SR176 application note

a. Reading serial number

ST SR176 is not ISO/IEC 14443-3 compliant, so it is not seen by the ID-Only template. Nevertheless, reading serial number is as easy as reading first 8 bytes from the memory :

- Set LOC.MEM = $_h$00
- Set TOF.MEM = $_h$02

b. Reading other data

ST SR176 is structured as 16 x 2-byte blocks (numbered 0 to 15). LOC.MEM specifies the block number (note that the data must be aligned on a block boundary). 8 blocks (i.e. 16 bytes) are always read by this template before formatting.

Make sure that 2 x LOC.MEM + length specified in TOF.MEM doesn’t exceed the actual capacity of the card, i.e. 32 bytes.

2.5.7. ASK CTS256 and CTS 512 application note

(to be written)

2.6. MIFARE CARD PROCESSING TEMPLATE

2.6.1. Lookup list

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

2.6.2. Output format

Name	Tag	Description	Size
TOF.MIF	$_h$t1	Mifare output format.	1

Same as Memory output format (2.5.2).

2.6.3. Output prefix

Name	Tag	Description	Size
PFX.MIF	$_h$t2	Mifare output prefix.	Var.

Same as ID-only output prefix (2.4.3).

2.6.4. 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, reader tries to read the block specified in LOC.MIF (16 bytes), and then truncate the data 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 definitively 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)[18] 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 the beginning of the first block marked with the specified AID.

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

2.6.5. 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 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

[18] 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 .

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 sector number as address
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 sector number as address
5 – 4	01	Diversify the key using RC171 algorithm *(see chapter 10)*
3 – 0		RFU

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 Memory 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[19] (1 to 64)

Default value : unspecified.

Values are MSB first.

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

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.

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 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
		Index of key in Desfire application
3 – 0	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 Memory 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[20] (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

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

a. T=CL options 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	Disable CID

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 enough to fetch the data.

Name	Tag	Description	Size
AU1.TCL	$_{h}$t5	TCL APDU 1	Var.

> ✋ Card's Status Word is checked. If SW is different than $_{h}$9xxx, answer is discarded.
>
> Reader's internal buffer is limited to 128 bytes. If card's answer is longer, it will be discarded.

2.8.7. T=CL APDU 2

Typically this is a Select File command.

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

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

> ✋ Card's Status Word is checked. If SW is different than $_{h}$9xxx, answer is discarded.
>
> Reader's internal buffer is limited to 128 bytes. If card's answer is longer, it will be discarded.

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. If SW is different than $_h$9xxx, answer is discarded.

Reader’s internal buffer is limited to 128 bytes. If card’s answer is longer, it will be discarded.

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 in 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 Memory 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[21] (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. If SW is different than ${}_{h}$9xxx, answer is discarded.

Reader's internal buffer is limited to 64 bytes. If card's answer is longer, it will be discarded.

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. If SW is different than ${}_{h}$9xxx, answer is discarded.

Reader's internal buffer is limited to 64 bytes. If card's answer is longer, it will be discarded.

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. If SW is different than ${}_{h}$9xxx, answer is discarded.

Reader's internal buffer is limited to 64 bytes. If card's answer is longer, it will be discarded.

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

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
PIN	$_{h}$6F	Console access PIN code

3. CONFIGURING IWM-K632

There are two ways to configure IWM-K632 :

- Using a Master Card, formatted with **iwmk632cfg** 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 the hardware, default factory settings for IWM-K632 firmware are :

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

Always configure IWM-K632 properly before installation as there are little chances that default configuration matches your requirements.

3.1. HARDWARE JUMPERS

Jumpers are available for basic configuration of the device.

3.1.1. RS-485, Wiegand and Dataclock hardware

Jumper	ON	OFF
1	**Normal mode**	Console mode
2	Red LED input disabled	Red LED input enabled
3	Green LED input disabled	Green LED input enabled
4	Buzzer enabled	Buzzer disabled

3.1.2. RS-232 and USB hardware

Jumper	ON	OFF
1	**Normal mode**	Console mode
2	Flash mode	**Normal mode** or console mode
3	Buzzer enabled	Buzzer disabled
4		

Switch JP3 allows selection between USB and RS-232 mode.

3.1.3. Note on console mode

The "console mode" jumper has three effects :

- Enable serial line access when the reader is configured for Dataclock or Wiegand operation (since serial line is multiplexed with Dataclock / Wiegand outputs, it is otherwise disabled),
- Force serial communication baudrate to 38400bps,
- Activate the echo on the serial line, and enable a few trace message for testing purpose.

> ☠ "console mode" inhibits normal operation of the reader.
>
> Do not forget to switch back "console mode" jumper to OFF after configuration.

3.2. CONNECTING IWM-K632 TO A COMPUTER

3.2.1. IWM-K632 dataclock / wiegand / RS-485

Use one of following Pro-Active interface to connect the reader (through its PC-Link Connector) to a Windows-based computer :

- INT-USB-232 (USB)
- INT-232 (RS-232)

If using INT-USB-232 (USB) interface, you'll have to install the *USB Virtual Serial Device* driver ("VCP" subdirectory under Pro-Active CSB Quickstart installation directory).

Use HyperTerminal or any equivalent terminal emulator to communicate with the reader[22].

3.2.2. IWM-K632 RS-232 / USB

Directly connect USB or serial interface to the computer. For USB reader, you'll have to install the *USB Virtual Serial Device* driver ("VCP" subdirectory under Pro-Active CSB Quickstart installation directory).

Use HyperTerminal or any equivalent terminal emulator to communicate with the reader[21].

[22] 38400bps, 8 data bits, 1 stop bit, no parity, no flow control

3.2.3. *Testing connection*

- Set “console mode” jumper to ON,
- Power-up (or reset) the reader,
- Reader sends its identification string :

```
Pro-Active K632 Reader [1.00]
```

3.3. RETRIEVING IWM-K632 INFORMATIONS

3.3.1. *Firmware version*

Enter “`ver`” to read IWM-K632 firmware version.

3.3.2. *Firmware configuration*

Enter “`sho`” to read IWM-K632 configuration.

3.4. ENABLING CONFIGURATION COMMANDS

IWM-K632 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`”)[23].

3.5. ACCESSING IWM-K632 CONFIGURATION

3.5.1. *Reading configuration tags*

Enter “`cfg`” to list all configuration tags.

[23] 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.

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.

3.5.2. *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).

3.5.3. *Writing keys in RC'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.

3.5.4. *Reading RC'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.

3.5.5. *Writing RC'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-K632 firmware (but it is the configuration vector for IWM-K531 firmware).

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

3.6. APPLYING NEW CONFIGURATION

New configuration is applied only after reset.

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

3.7. 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

This chapter refers to RS-232, RS-485 and USB versions of IWM-K632.

4.1. SERIAL FRAME MARKERS

Serial frame markers are configured by bits 7-6 of SER .

4.1.1. *When addressing is disabled*

Consider data ’01 23 45 67’,

- If bits 7-6 = $_b$00, frame is “01234567”.
- If bits 7-6 = $_b$01, frame is “<BEL>01234567<CR><LF>” where <BEL> is the ASCII bell (or ring) character ($_h$07), <CR> the ASCII carriage return ($_h$0D), and <LF> the ASCII line feed ($_h$0A).
- If bits 7-6 = $_b$10, frame is “<STX>01234567<ETX>” where <STX> is the ASCII “start of text” character ($_h$02), and <ETX> the ASCII “end of text” ($_h$03).
- If bits 7-6 = $_b$11, frame is “<BEL><STX>01234567<ETX><CR><LF>”.

4.1.2. *When addressing is enabled*

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

- If bits 7-6 = $_b$00, frame is “a>01234567”.
- If bits 7-6 = $_b$01, frame is “<BEL>a>01234567<CR><LF>”.
- If bits 7-6 = $_b$10, frame is “<SOH>a><STX>01234567<ETX>” where <SOH> is the ASCII “start of header” character ($_h$01).
- If bits 7-6 = $_b$11, frame is “<BEL><SOH>a><STX>01234567<ETX><CR><LF>”.

4.2. SERIAL INPUT

IWM-K632 accepts short commands from the host, to drive LEDs and buzzer output mostly.

IWM-K632 doesn’t echo received data (unless “console mode” jumper is ON).

If received command has been understood by IWM-K632, it replies with <ACK> before executing the requested action. Otherwise, it replies with <NACK>.

4.2.1. Addressing disabled

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

4.2.2. Addressing enabled

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

4.2.3. 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

✋ Set jumpers appropriately, and choose proper configuration in CLD and CBZ to allow device to control its LEDs and/or its buzzer.

5. WIEGAND APPLICATION NOTE

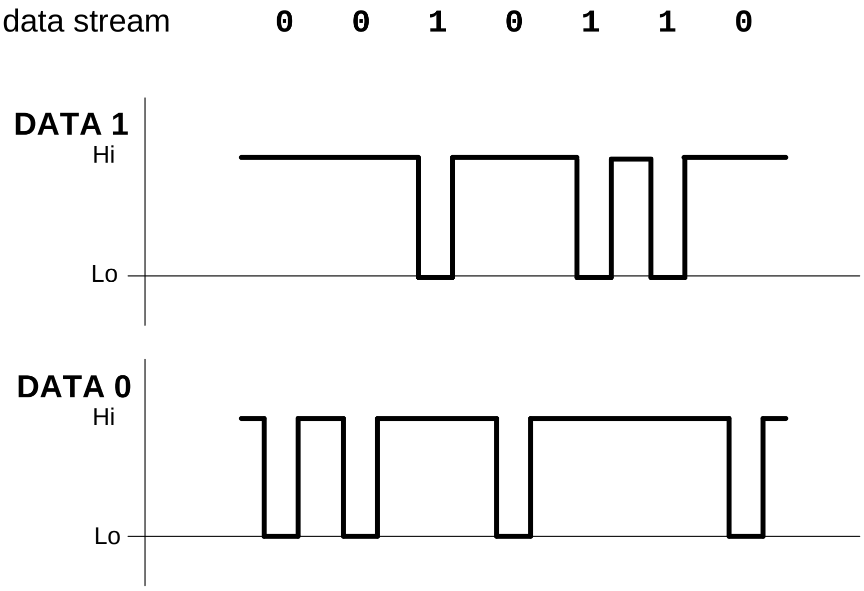
5.1. THE WIEGAND INTERFACE

5.1.1. *Bit format*

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.

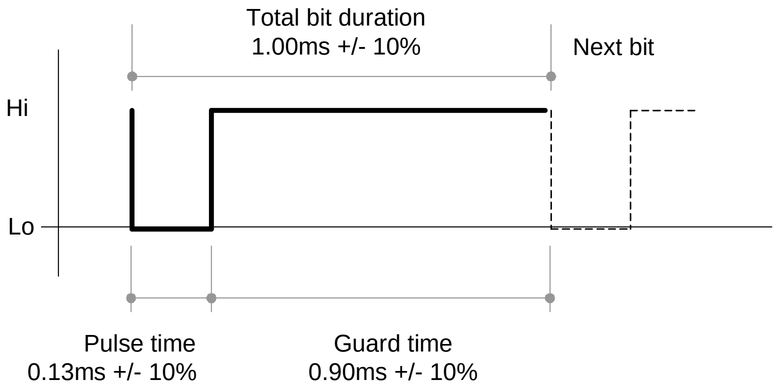


a. *Electrical levels*

	Level to GND
Output level high	4.0V min, 5.5V max
Output level low	0.0V min, 1.0V max

✋ DATA0 and DATA1 are not open collector output. Internal pull-up resistors are included in the reader.

b. Timings



Those are the default timings. They can be altered by writing in WGD.

5.1.2. Frame format

Wiegand output format is driven by configuration data.

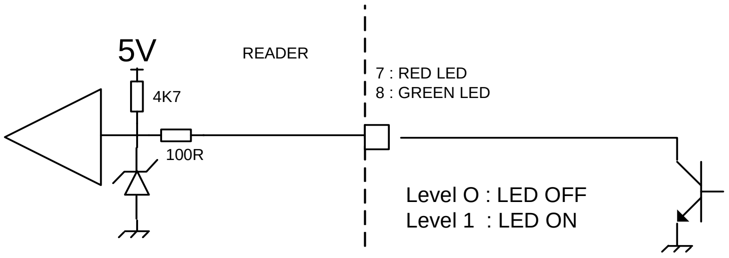
Please refer to chapter 2.2.4 for details.

5.2. LED INTERFACE

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

	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.



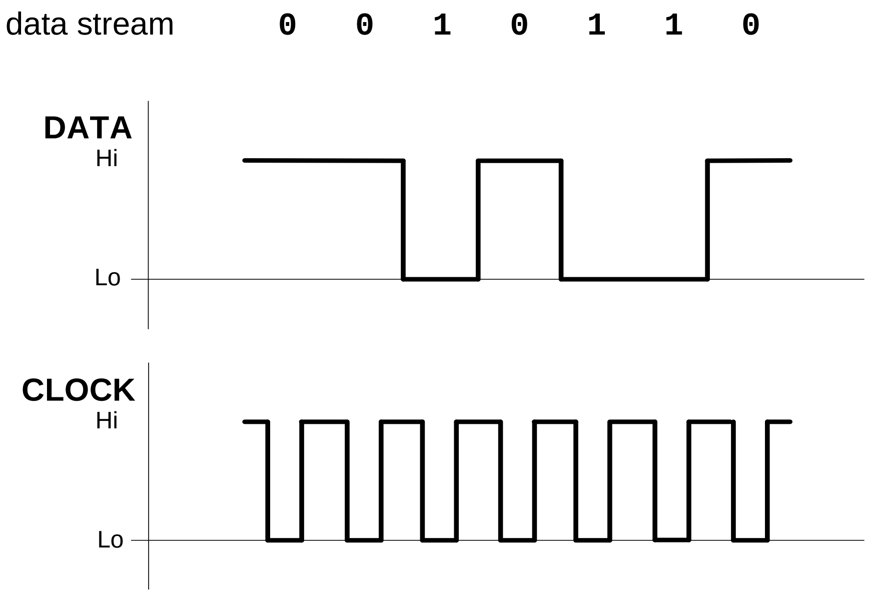
Set jumpers appropriately, and choose proper configuration in CLD to enable LED inputs.

6. DATACLOCK APPLICATION NOTE

6.1. THE DATACLOCK INTERFACE

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).

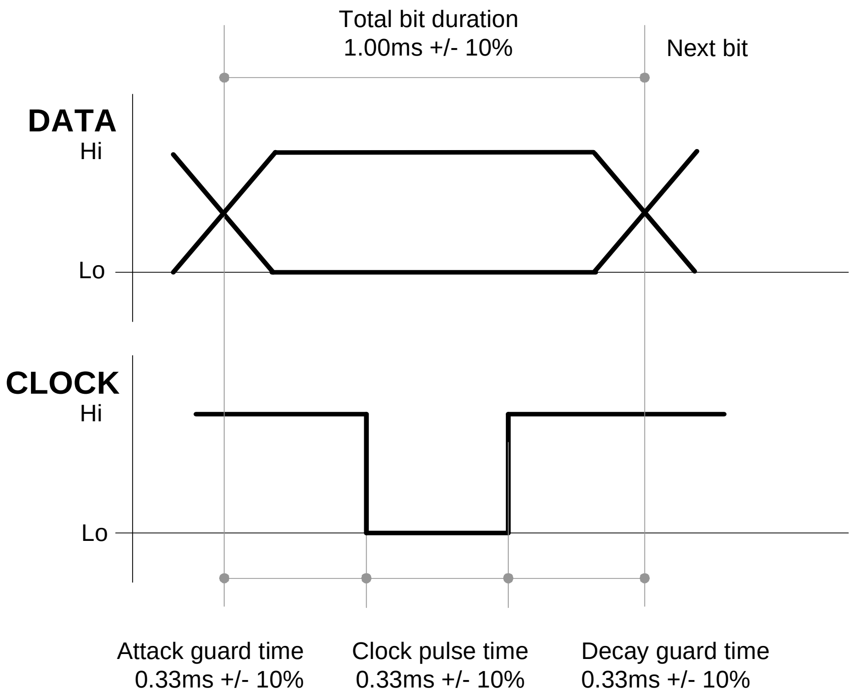


a. *Electrical levels*

	Level to GND
Output level high	4.0V min, 5.5V max
Output level low	0.0V min, 1.0V max

DATA and CLOCK are not open collector output. Internal pull-up resistors are included in the reader.

b. Timings



Those are the default timings. They can be altered by writing in DTC.

6.1.2. 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*
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

Value	*Bit pattern*	*Reserved for*
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 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 ${}_{d}$15) 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 ${}_{d}0$ (${}_{h}0$) to ${}_{d}15$ (${}_{h}F$) being allowed.

6.3.2. Frame prefix and postfix

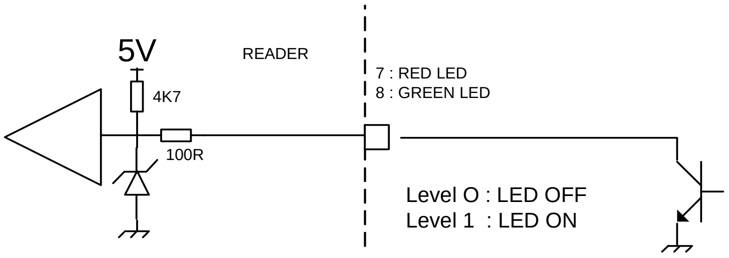
RAW frames are transmitted without prefix and postfix.

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.



✋ Set jumpers appropriately, and choose proper configuration in CLD to enable LED inputs.

7. SPECIFICATION OF MASTER CARDS

> This chapter is provided as a mean for security experts to evaluate IWM-K632 Master Card architecture.
>
> Customers do not need to implement this part by themselves, since **iwmk632cfg** 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,
- 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, reader checks that data comes with a valid digital signature. The signing key can't be changed, and is only known to 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 **iwmk632cfg**– 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
		Index of key in Desfire application
3 – 0	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 **iwmk632cfg**– is able to sign the Master Cards[24].

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		*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	0000	*Those bits are RFU and must be 00*

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

[24] 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[25],
- Let *SignKey* be the 16-byte key,
- Diversify *SignKey* from card's UID, using HMAC-MD5 diversification algorithm[26] to get *DivKey*,
- Compute *Sign* = HMAC-MD5 (*Block*) using *DivKey* [27].

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

[25] 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.

[26] See 8.3.1

[27] See 8.2

8. USING IWMK632 SOFTWARE TO CREATE MASTER CARDS

8.1. OVERVIEW

iwmk632cfg is a command line software (running on Microsoft Windows) to create Master Cards. **iwmk632cfg** needs a Pro-Active CSB4 (S or U) contactless coupler to program the cards.

Enter **iwmk632cfg -h** to read the complete list of command line switches and options, and the complete list of sections and variables for configuration files.

iwmk632cfg software comes with various sample configuration files that shows typical configurations of IWM-K632.

Master Cards are NXP Desfire 4k.

8.2. CONFIGURATION FILES

iwmk632cfg uses a configuration file to retrieve configuration data to be written into the Master Card.

Configuration files are made like standard Windows "INI" files. They can be edited using Notepad or any other text editor.

Each line in one 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-K632 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

- Create your configuration file and save it in the same directory where **iwmk632cfg** is installed, for instance with the name *siteconf.ini*.
- Open a new shell (on Windows : Start Menu → Run → cmd.exe),
- Go to the directory (command "cd") where iwmk632cfg is installed,
- Plug and power-on your CSB4,
- Put a virgin Desfire card on the CSB4,
- Enter **iwmk632cfg –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-K632 are shipped 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[28].

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…

[28] That's because *1st level Master Card* has got the authentication key written in it, and anybody may retrieve it using **iwmk632cfg** software, where the authentication key is only used to secure *2nd level Master Cards* and 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[29],

- Put a virgin card on the CSB, label it "1st level Master Card",
- Enter **iwmk632cfg –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 CSB, label it "2nd level Master Card",
- Enter **iwmk632cfg –c *siteconf.ini*** ,
- Use this Master Card to write complete configuration in the reader(s).

[29] 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 CSB, label it "Erase all Master Card",
- Enter **iwmk632cfg –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 enough 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

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

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's a kind of digital signature[30].

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*.

[30] 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.

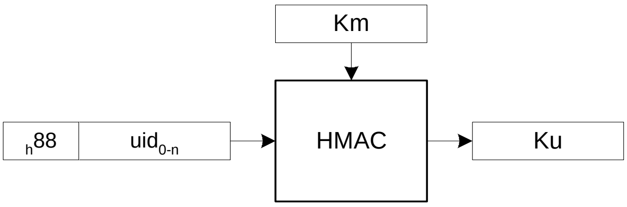
9.3.1. DES & Triple-DES key diversification algorithm

The algorithm takes as inputs :

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

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.

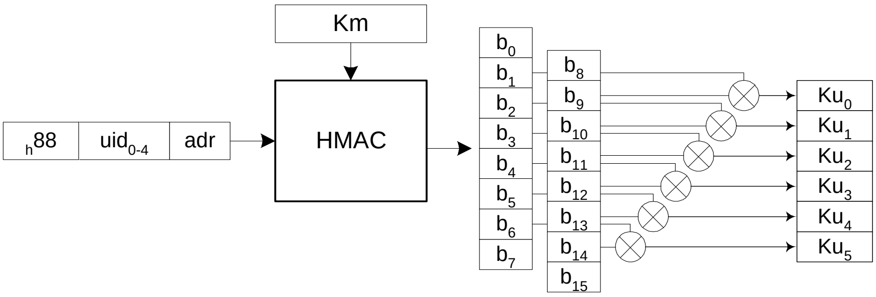
9.3.2. Mifare key diversification algorithm

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.

[31] 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.

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.

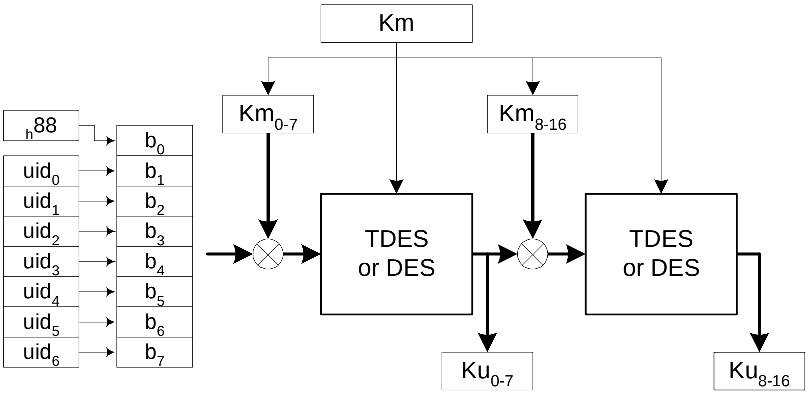
The algorithm takes as inputs :

- A 16-byte Triple-DES master key (Km)[32]
- 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.

[32] 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

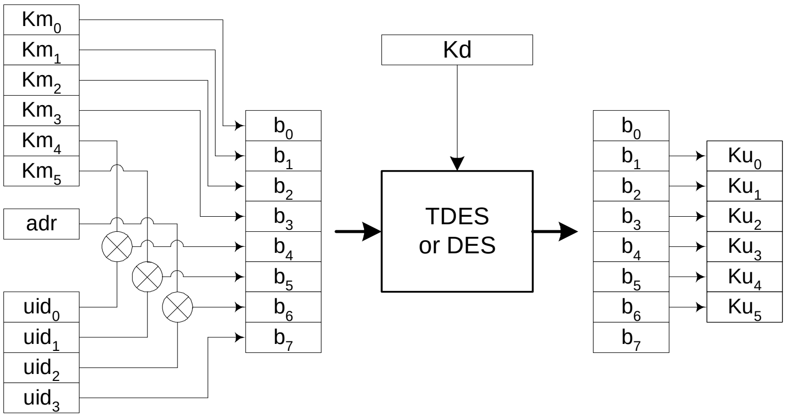
The algorithm takes as inputs :

- A 6-byte master key (Km)
- A 16-byte Triple-DES diversification key (Kd)[33]
- 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 the block to be read.

[33] 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 feed 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 722 C

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