Hybrid Card Controller

“862” Series

and derived models

RS232 or USB

Command reference manual

Protocol description V2.01

Nov. 2007

ddm hopt+schuler
Königsberger Strasse 12
D-78628 Rottweil

+49 741 / 26 07 - 0
+49 741 / 1 33 98
www.hopt-schuler.com

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing.

History

Date	Rev	Notes
2002-07-22	1.01	Initial draft
2002-09-30	1.02	Preliminary
2002-12-30	1.03	For Reader 862 V1.xx with 1st version of BOOT modules
2003-07-02	1.04	For Reader 862 and 867 - 2nd version of Loader/Boot module
2003-10-30	1.05	Includes Synchronous cards interface
2004-02-20	1.10	Motorized readers: 855 and 869: additional functionalities
2004-04-20	1.11	Model 868: additional descriptions
2004-11-24	1.12	Model USB: additional descriptions / Electrical connection / More photos
2004-12-16	1.2	ICC Activation: more parameters / Activation outside EMV2000 scope
2005-11-16	2.0	Additional bits in Status words Function “ARM”: response depending on configuration ICC Activation: updated Additional maintenance functions for motorized models This specifications applies to all readers with firmware version > 2.0
2007-11-07	2.01	Model 855 USB: additional descriptions / Electrical connection

Table of contents:

I.Introduction

This document describes the requirements, operations, and usage of the smart card / magnetic stripe readers, models 862 and compatibles

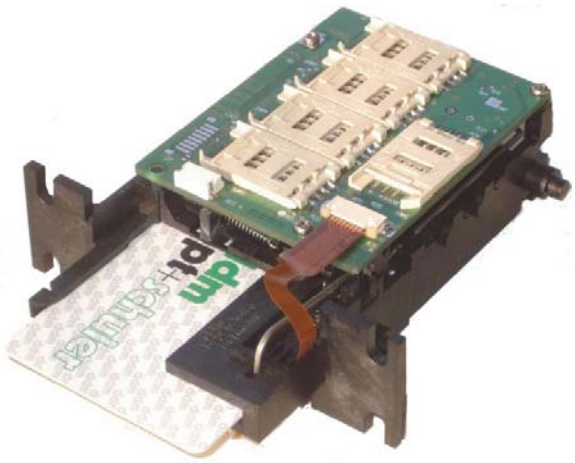
The models 862, 867, 868, 845 … are manually operated insert readers

The models 855 and 869 are motorized readers

The purpose of this manual is to describe the information exchanged between the card reader and an attached host system.

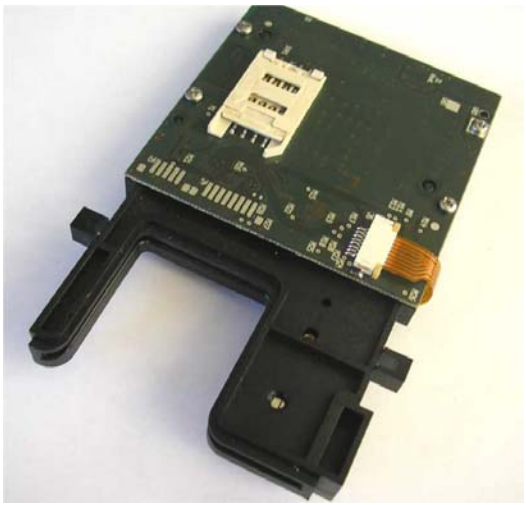
It's destined to developers who are familiar with technical terms in the domain of magnetic card and/or smart card applications.

We strongly recommend reading the appropriate ISO/IEC standards, especially ISO/IEC 7811 for magnetic cards, and ISO/IEC 7816 for smart card applications.



Model "862"

Model "867"



Model "868"

Model "855/869"

1. Properties

All these Models are microprocessor controlled, card reader with a serial interface.

- Models 862, 867, 845 and 868 are manually operated insertion type.
- Models 855 (and 869) are motorized type
- Reads and decodes up to three tracks of magnetic data (F2F) simultaneously (except models 845 and 869 that are chip-only models).
- Reads and writes a limited number of synchronous cards (memory cards): type SLE 4432/4442; SLE 4428 or GPM8K, I²C cards with 8-bits and 16-bits address
- Communicates with asynchronous cards (microprocessor cards) using T=0 or T=1 protocol according to ISO/IEC 7816. Five-volt and three-volt cards are supported.
- Includes up to 5 SAM (Security Application Modules)
- "EMV 2000 – level 1"compliant, for payment system applications
- The reader's firmware is downloadable through the serial port and is stored in non-volatile Flash memory.
- Optional locking system can be added (except on model 868) to lock the card mechanically in seated position, and secure chip card transmission. Locking and unlocking is fully host controlled.
- Optional power fails detection to automatically unlock the card in case of a power failure. This requires a capacitor that has to be soldered on top of the reader's PCB or connected through a header on the board.
- Optional external LED port for two (or one dual color) LED available, fully host controlled through commands. The LED can be connected through a header on the board.
- Software drivers available for Windows environment (98, 2000, XP) and Linux.
- Demo software available for Microsoft Windows ™ environment and SuSE Linux.
- Very low power consumption:

  Models 862 / 867 / 868:

  From 6V to 12V, approx. 30 mA nominal

  250 mA maxi, during 110 ms, at latching operation

  Models 855 / 869:

  12 V, approx. 90 mA nominal

  160 mA when motor is running

  250 mA when the shutter is opened
- The USB models are bus powered, and do not need additional power supply, except for the 855 and 869 versions.

2. Electrical connections

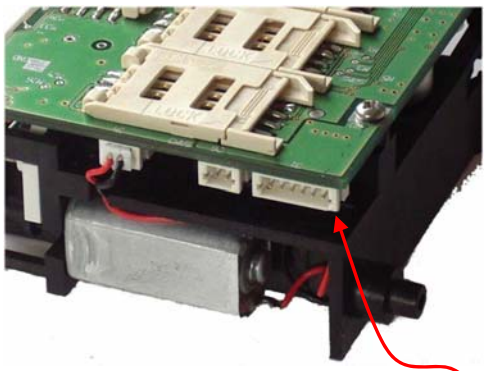
RS232 models

One 6-pins connector, at the back of the device, is used to connect the device to a host computer and to the power supply:

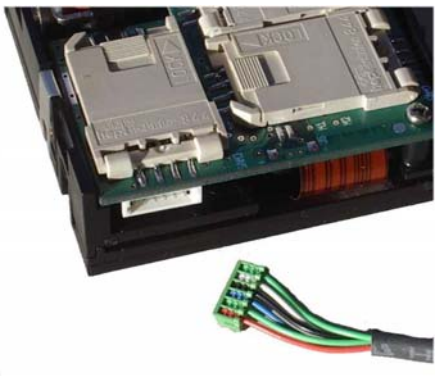
- The counterpart to the connector on the reader side is either
  - ZH connector series, part number ZHR-6 or
  - ZR connector series, part number 06ZR-8M
  Both are from manufacturer JST (http://www.jst.com/)
- Power supply: between 6 and 12 Volts DC (reader with manual insertion)
- Power supply: 12 Volts DC (for motorized reader insertion)
- RS232 – standard V24 serial link (default baud rate: 38400 b/s)

The pinning is:

On Reader	JST pining	direction	DB9 pining	RS232 from PC
CTS	1	←	4 (or 7)	DTR (or RTS)
TXD	2	→	2	RXD
DTR	3	→	8 (or 6)	CTS (or DSR)
RXD	4	←	3	TXD
GND	5	⇔	5	GND + 0V
Power	6	←	Power supply	



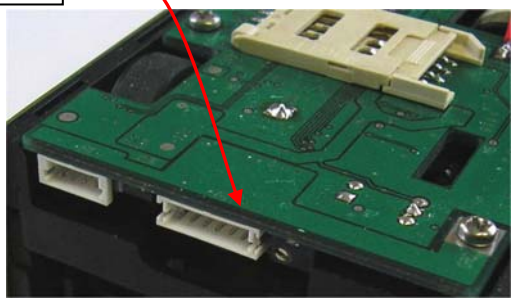
Model “862 serial”: con J1



Model “867” serial: con J1

Pin 1

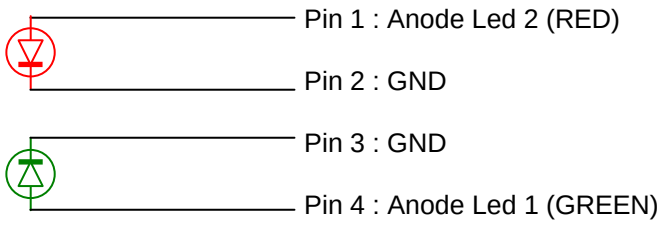
Model “868 serial”: con J1

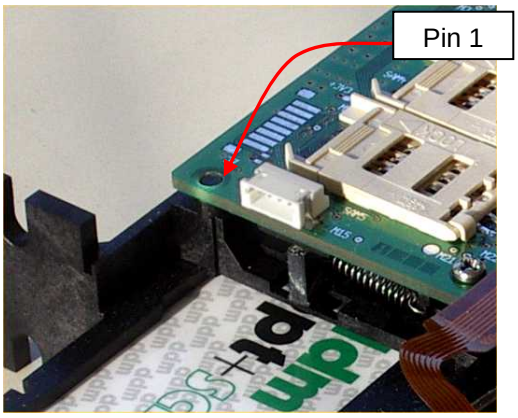


Model “855 serial”: con J1

Connector for additional LEDS

On most models, a 4-pin JST connector permits the connection with 1 or 2 LEDS





On model 862 serial: Con J5
On model 862 USB: Con J3

On model 868serial: Con J2
On model 855 / 869: Con J2

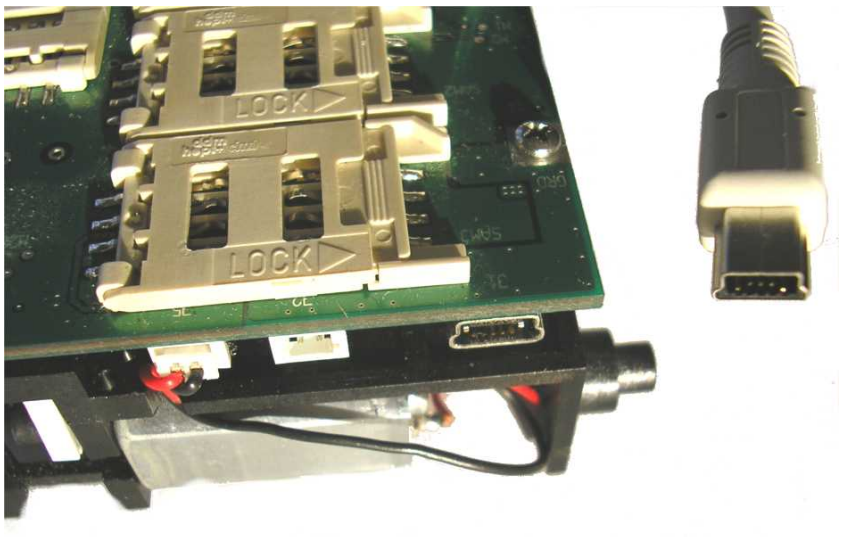
Note:

There's no LEDS connector on the old 867 model, because led 1 is directly solded onto the PCB

USB models

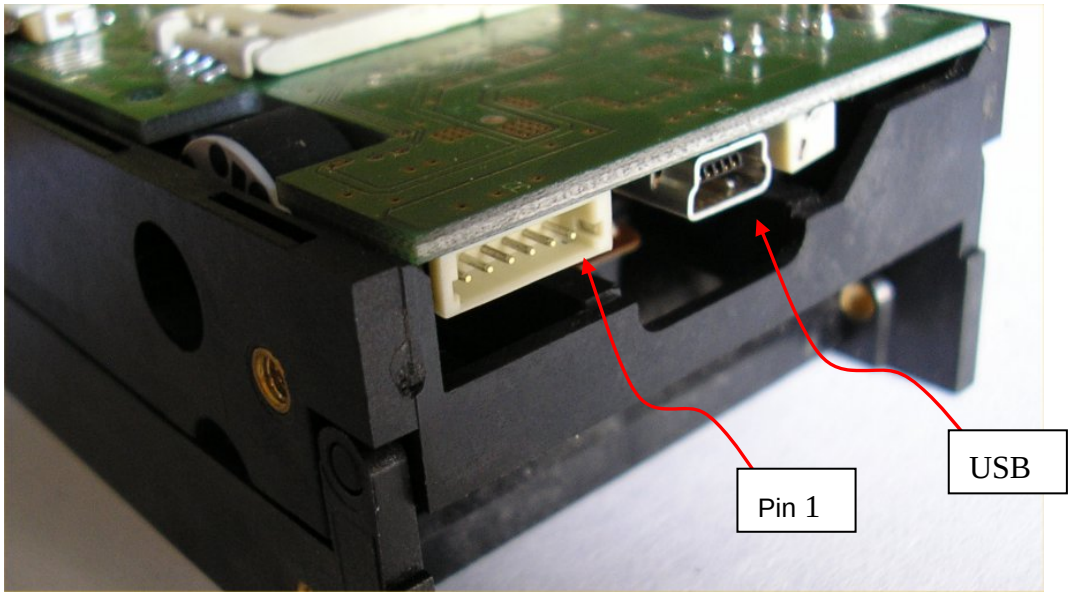
Manual readers (without motor) don't need additional power-supply; they are bus-powered.
Only the motorized model 855usb needs an additional 12V for the card transport motor
These readers are all equipped with the mini USB connector

- Ref "Mini USB – type B"



The 855 USB is equipped with a JST 6-pins connector for

JST pining	Function
1	External 12 V power supply
2	GND (0 Volt)
3	Green Led Anode (+)
4	Green Led Cathode (-)
5	Red Led Cathode (-)
6	Red Led Anode (+)



II. Protocol descriptions

This chapter describes, in summary, the protocol specifications.

Commands and data are all exchanged, with the host, through the serial link RS232

The USB readers are connected through virtual COM port. So their connection properties could be see as standard RS232 ones

This protocol, called **USI 2**, allows all actions the reader will be able to perform.

1. Transmission format

The default parameter settings are:

Format: 1 start bit, 8 data bits, no parity

Rate: 38400 baud

No DTR/CTS control.

Address: 00_h

These parameters can be changed, depending on settings from the configuration data. See chapter "Reader's configuration data" to set the appropriate parameters.

Baud rate:

Standard values, from 2400 baud to 38400 baud, can be set as transmission speed.

Parity:

3 possible values: no parity, even parity, or odd parity

Reader's DTR driving:

If used, this signal has to be connected on the host's CTS or DSR input

If this mode is inactive, the signal DTR is set when the application starts, and cleared when the application exits.

If this mode is active, the signal DTR is cleared during the host's request execution. In fact, the reader is not able to read a new request from the host, when the process commanded by the previous command, is still running. The host can check this signal to know if the reader is ready or not to receive a new request.

Reader's CTS control:

If used, this signal has to be connected to the host's DTR or RTS output

If this mode is inactive, the reader does not take care of this signal, and it sends the data regardless whether the host is ready to receive or not.

If this mode is active, the reader checks this signal each time it has to send a frame, just to know if the host is ready or not to receive this frame. If the CTS signal is active, the frame is sent immediately. Otherwise the reader is blocked and will wait until the CTS signal will become active.

Reader's address:

The default address is 00_h.

In a network configuration, for example if many readers connected to the host via a RS422 connection, each reader can be configured with a different address (between 1 to 127). So the host has to specify the address of the reader it wants to request.

2. Frame structure

The transmission format is conformed to V24 specifications

All frames are structured as followed:

SOH	ADDR	LEN	DATA	BCC

Or

SOH	ADDR	0000_h	DATA	EOT	BCC

With:

SOH = 01_h, EOT = 04_h

ADDR:

This field (one byte) is the device address.

Basically, this value is null (00_h). For extended use, i.e. in a network configuration, a no-null value should be set. This field is the address of the reader that might be concerned by this request.

This address must be set in the reader's configuration data.

LEN:

This field is encoded in two bytes (MSB first), and gives the length of the DATA part. The allowed value is between 1 and 264 (coded 0001_h to 0108_h)

When this value is null, the data part is generally encoded in ASCII mode. The EOT control character signals the end of the Data

When this value is not null, it is the data length, and no EOT control character is present.

DATA

This is the command or the response message. The next chapter defines it.

BCC:

Is the "Block Check Character". Its value is equal to the result of exclusive OR of all preceding bytes (SOH and EOT bytes are included).

A maximum of 200 ms is allowed between two consecutive characters.

III. Messages descriptions

This chapter describes the part DATA from frames transmitted between the reader and the host.
Theses specifications are the same for RS232 transmission and USB transmission

1. Command descriptions

3.1.1. Card handling

Card position

cCARD_POSITION

This request is used to know the current card position. The reader has got two switches, one at the front position to detect a card, and the second one at the rear to know if the card is completely inserted.

Action:

- Return one character depending on the switches status
  - '**p**' (70_h) when no card is detected
  - '**q**' (71_h) when a card is detected at front position
  - '**s**' (73_h) when a card is completely inserted.

Note: The two less significant bits of the response character indicate the switches status
b0: Start switch (at front position)
b0 = 0 when no detection b0 = 1 when card is detected
b1: End switch (at rear position)
b1 = 0 when no detection b1 = 1 when card is inserted

If the "Auto switch report" mode is active, the position character is automatically sent to the host, each time a card position change is detected. This mode is set into the reader's configuration data. (See this chapter to know how to set this mode)

Card locking

cCARD_LOCK

When the reader is equipped with a card locking system, this command is used to activate it.

Action:

- Activate the locking motor to **lock** the card
- Transmit a **rACK** as acknowledge, if the locking status indication is ON; or transmit a **rERROR** if the requested action could not be satisfied.

With the model 862, the locking command can only be processed when a card has been completely inserted. If not, a **rINVALID** is returned.

With the model 867, the card is mechanically (and automatically) locked when the user push the card into the reader

With the motorized models 855 and 869, the locking command consists to move the card into the reader at the 'chip' (or 'inside') position (when a card is still present in front of the reader)

With the models (like 868 or 845) without locking system, this function is not defined.

Card unlocking

cCARD_UNLOCK

This command has to be used to return the card to the user

Action:

- Deactivate the locking system. With motorized reader, the card is moved to front position
- Transmit a **rACK** as acknowledge, if the unlocking status indication is OK; or transmit a **rERROR** if the requested action could not be satisfied.

Note: The Unlocking command is refused, while the chip card is activated (powered on). In this case, the code **rINVALID** will be returned.

Note2: With models 855 / 869, magnetic decoding is performed during card moving, on tracks that have not been correctly decoded during the card insertion phase.

Card Capture

cCARD_CAPTURE

With motorized models, this command has to be used when the card should not be returned to the user. The card will be captured by the system, i.e. the card is ejected to the rear side.

Action:

- Move the card to the rear direction, until the card is ejected through the rear side
- Transmit a **rACK** as acknowledge, when no more card is detected by the reader. If the execution fails (card jam), the code **rERROR** is returned

Note: This command is only available on motorized readers (model 855/869).
With the other models, the error code **rINVALID** will be returned.

3.1.2. Device information

Reader status word

cDEV_STATUS

This command makes the reader to return its status word. Depending on models, up to 4 bytes are returned.

Action:

The readers transmits <*STATUS1*> + <*STATUS2*> + <*STATUS3*> + <*STATUS4*>, with

<*STATUS1*>:

Bit 0:	0: If no card is detected	1: if a card is detected
Bit 1:	0: If no card is inserted	1: if a card is completely inserted
Bit 2:	0: If card is not locked	1: if card is locked
Bit 3:	0: ICC/SAM is inactive	1: ICC/SAM is activated (powered ON)
Bit 4:	0: Non auto switch report	1: automatic switch report mode
Bit 5:	0: No magnet stripe	1: magnet stripe has been detected
Bit 6:	0: Idle mode (not armed)	1: armed mode
Bit 7:	0: No-auto ATR mode	1: auto ATR sending mode

<*STATUS2*>:

Bits 1-0: Green led state: 00: led is off; 01: led is on; 10: led is flashing
Bits 3-2: Red led state: 00: led is off; 01: led is on; 10: led is flashing
Bit 4: Arm mode active in “debug” mode
Bit 5: Arm mode active, with magnetic “Pre-head” detection
Bits 6 & 7: not defined (always null)

<*STATUS3*>:

Bits 0 to 2: Current ICC/SAM selection (0 = User card; 1 to 5: SAM id.)

Bit 3:	0: If no SAM#1 is present	1: SAM#1 present in its socket
Bit 4:	0: If no SAM#2 is present	1: SAM#2 present in its socket
Bit 5:	0: If no SAM#3 is present	1: SAM#3 present in its socket
Bit 6:	0: If no SAM#4 is present	1: SAM#4 present in its socket
Bit 7:	0: If no SAM#5 is present	1: SAM#5 present in its socket

<*STATUS4*>:

It's the error code of the last ICC data exchange (with asynchronous card)

Bit 0:	1: Electrical abnormality
Bit 1:	1: Timeout in ATR reception
Bit 2:	1: Character reception timeout
Bit 3:	1: Parity error during reception
Bit 4:	1: Collision detection during transmission
Bit 5:	1: Parity error during transmission
Bit 6:	1: Error in protocol T=0
Bit 7:	1: Error in protocol T=1

Reader reset/restart

`cDEV_RESET`

This command makes the device restart, and perform all initialization function. When the reader restarts, it sends the frame "**`rRESTART`**".

Note: The restart procedure takes needs some times to process. The response frame needs a delay between 1 and 1.5 seconds, after the reset command had been sent.

Version and description strings

`cDEV_DESCRIPTORS + <N>`

This command is one or two character length. The returned message is:

When parameter **`<N>`** is not present

The reader returns the application **firmware** version information

When **`<N>`** = '0' (30_h or 00_h)

The reader returns its **serial number**

When **`<N>`** = '1' (31_h or 01_h)

The reader returns "product – version – copyright" information

When **`<N>`** = '2' (32_h or 02_h)

The reader returns the reader **model** reference

When **`<N>`** = '3' (33_h or 03_h)

The reader returns the **PCB** (Printed circuit board) reference.

When **`<N>`** = '4' (34_h or 04_h)

The reader returns the **application** firmware version

When **`<N>`** = '5' (35_h or 05_h)

The reader returns information about the **boot** system program.

When **`<N>`** = '6' (36_h or 06_h)

The reader returns information about the **loader** module, which has to be activated by the boot program, to perform flash erasing/writing.

When **`<N>`** = '7' (37_h or 07_h)

The reader returns the reader's current configuration data block (128 bytes).

When **`<N>`** = '8' (38_h or 08_h)

The reader returns version information about the **USB library** system. (Only available on USB model)

3.1.3. LED's

2 LED's are available to the host's application: The first one is defined as green, the second one as red.

Default state (when the reader starts) is "off"

The following commands have to be used to drive them.

Fixed states

cLED_GREEN_ON – cLED_RED_ON
cLED_GREEN_OFF – cLED_RED_OFF

Action:

- Turn the specified led ON or OFF
- Transmit a **rACK** as acknowledge

Flashing effects

cLED_GREEN_FLASHING – cLED_RED_FLASHING

Action:

- Begin flashing the specified led on and off
- Transmit a **rACK** as acknowledge

Notes: If both LEDs are set in flashing mode, and the reader is equipped with a bicolor led, the given effect will be a switching between the two colors: green and red. (When one color is off, the other one is on)

Notes: With devices equipped with only one simple LED, only the commands **cLED_GREEN_*XXXX*** are allowed

3.1.4. Magnetic decoder

The purpose of the ARM command is to prepare the reader waiting for a new magnetic card. This reader status is herein called "ARM mode". When a card is being moved through the reader, magnetic decoding is performed. Data are analyzed, and the reader is then able to return the valid (or invalid) magnetic stripe data. At this state, the reader is exiting the "ARM" mode.

Depending on the reader's configuration, the option "***Arm-And-Ack***" makes the reader automatically inform the host when it leaves the ARM mode and data are available. This option is useful to avoid a reader polling.

When the option "***Arm-And-Ack***" is not activated, the reader does not transmit, by itself, any message when the ARM mode is done. So this principle requires the host to poll the reader (using for example the **cDEV_STATUS** command) to know about a status change on the reader.

Arm to read (with manual model 862 / 867 / 868)

cARM

This command prepares the card-device to read the magnetic tracks. All previous data are cleared.

Action:

- Activate the "arm" mode. All previous data (concerning magnetic tracks) are deleted
- Transmit a **rACK** as acknowledge
- The reader waits for card insertion/withdrawal

Then a new card is introduced into the reader – or – The present card is withdrawn

When the option "***Arm-And-Ack***" is activated, the following sequence applies just after a card has been moved.

- The reader automatically sends a second **rACK** when the magnetic decoding process has been done, after the card has been inserted or withdrawn.
- Or, if no magnet stripe has been detected when the card is completely inserted/withdrawn, the message **rNO_MAG_CARD** is returned, instead of **rACK**
- This second transmitted code indicates that the reader is exiting the arm mode

Note: If an ARM command is sent when the reader is already in arm mode, then the code **rERROR** will be returned as response

Arm to read (with motorized model 855 / 869)

cARM
cARM + <P>

This command makes the device ready to accept a new card. Previous data are flushed. When a card is detected at front position, the reader will start its motor to intake it. Decoding is performed during this insertion phase.

If a card is still inside the reader, and some tracks have not been successfully decoded, then the card will be moved out, and immediately move in again, just to pass the card under the head and to perform another magnetic reading again. Each time, the card moves under the magnetic head (also with command **cLOCK** or **cUNLOCK**), no correctly decoded tracks will be read again.

When all tracks have been correctly decoded, the card will be no more moved with using this ARM command.

The parameter <P> is optional.

<P> = '0' (00_h or 30_h) or is missing: The pre-head detection is not activated

<P> = '1' (01_h or 31_h): The reader waits for a magnetic signal, to be detected by the pre-head, before opening the shutter. This option could only be activated, when the reader is equipped with a shutter and the optional (magnetic detector) pre-head.

Action:

- Activate the "arm" mode: The reader becomes ready to perform magnetic reading
- Transmit a **rACK** as acknowledge
- If a card is already into the reader, it could be moved to restart the reading process
- If no card is present, the reader waits for a new card insertion

A card could now be introduced into the reader.

When the option "***Arm-And-Ack***" is activated, the following sequence applies just after a card has been inserted into the reader.

- Send a **rACK** when the magnetic decoding process has been done, and the card has been successfully moved to the 'chip' position (inside the reader).
- In the case of a card jam, or if the shutter (if it exists) has not been completely closed, the code **rERROR** will be returned instead.
- If no magnet stripe has been detected when the reading process is done, the code **rNO_MAG_CARD** is returned instead

Arm to read (in debug mode)

cARM_DEBUG

This command is similar to **cARM**, except that an additional message is sent; exactly at the moment the reader is starting the magnetic stripe decoding.

This command doesn't take care about the "***Arm-And-Ack***" option

Action:

- Activate the "arm" mode
- Transmit a **rACK** as acknowledge
- Wait for card insert/withdrawal.

When a card is being moved in (or out) the reader

- Send a **rMAG_DETECT_ON** when a magnetic stripe is to be detected
- Send an **rMAG_DETECT_OFF** when the magnetic signal stops. This message signals the "arm mode" end.
- If no magnet stripe has been detected when the card is completely inserted/ withdrawn, the unique message **rNO_MAG_CARD** is returned, instead of **rMAG_DETECT_ON/OFF**

Arm Abort

cABORT

This command makes the reader to abort the 'ARM mode. No future decoding work will be done.

Action:

- Exit the arm mode. The reader returns into the idle state
- Transmit a **rACK** as acknowledge

3.1.5. Magnetic tracks: decoded data

Get data from standard ISO coded tracks

cMAG_ISO_T1
cMAG_ISO_T2
cMAG_ISO_T3

This command performs the ISO decoding from the bits read from the requested track, and then returns the response into an ASCII format.

Standard ISO format specifications are:

- Track 1: 210 bpi; 6 bits per char + odd parity bit. Length < 79 char.
  Start char is '%' (45_h); End char is '?' ($1F_h$)
- Track 2: 75 bpi; 4 bits per char + odd parity bit. Length < 40 char.
  Start char is ';' ($0B_h$); End char is '?' ($1F_h$)
- Track 3: 210 bpi; 4 bits per char, odd parity bit. Length < 107 char.
  Start char is ';' ($0B_h$); End char is '?' ($1F_h$)

Action

- If there's no decoding error, the data are returned as an ASCII string (without Start, End, and LRC characters)
- If there's a parity or a LRC error, or if no end character has been found, the response returned is **rERROR**
- If no magnetic stripe has been detected, or if no start character has been found (i.e. when the track is not encoded), the response returned is **rNO_DATA**

Get ISO data with given format

cMAG_FMT_T1 + <F>
cMAG_FMT_T2 + <F>
cMAG_FMT_T3 + <F>

This command is two char length. It has to perform the ISO decoding from the bits read from the requested track **T**, conformed to ISO track format **<F>**

The requested format **<F>** is respectively coded '1', '2', or '3' (01_h or 31_h, 02_h or 32_h, and 03_h or 33_h) for ISO1, ISO2, or ISO3.

These commands allow a track to be coded with a different ISO format. For example, if track 1 is coded like a standard track 3 (i.e. 4 bits + parity per char), the command should be:

cMAG_FMT_T1 + '3'

Action:

- The different returned codes are similar to the case **cMAG_ISO_T*i***

Get corrupted ISO data

cMAG_ERROR

When the previous ISO track reading command returns an error because of decoding parity and/or LRC errors, this command can be used to get the data regardless of the errors.

Action:

- The reader returns the incorrect (or correct) data as an ASCII string, conformed to the previous requested format.
- If there's no data because the track was not encoded, the value **rNO_DATA** is returned.
- If the previous request was **cMAG_CUSTOM_Ti**, the response **rERROR** is returned

Get data from tracks with custom format

cMAG_CUSTOM_T1 + <*F*>
cMAG_CUSTOM_T2 + <*F*>
cMAG_CUSTOM_T3 + <*F*>

This command is two char length. It has to perform a string decoding from the bits read, with each magnetic character being **<*F*>** bits length

The value **<*F*>** shall be comprised between '3' (03_h or 33_h) and '8' (08_h or 38_h).

The returned string is coded in binary mode.

No decoding

Action:

- Process data for the specified track, based upon the "number of bits" character
- Transmit the data formatted as one F-bits data word per returned character.
  The first bit is always a bit "one" (1); this bit initialize the first data of the frame.
  All the last null words will be skipped.
  Each data word is coded "low significant bit" (LSB) first.
- Note: With this request, no decoding verification can be performed. The user will have to implement its own control characters, and data protection codes.

Example:

If <F> = 3, the returned values V are: $00_h < V < 07_h$

If <F> = 5, the returned values V are: $00_h < V < 1F_h$

If <F> = 7, the returned values V are: $00_h < V < 7F_h$

Example:

When the decoding bits are:

000000000 **1**100 1001 1000 0000 0001 0101 1010 **0100** 0000 0000000

And <F> = 4 bits per character, the returned data frame will be:

03_h 09_h 01_h 00_h 08_h $0A_h$ 02_h

3.1.6. Chip card driver

This chapter defines the functions to be use to communication with asynchronous smart card or SAM (Secure application modules). The connections with them guarantee the respect of electrical specifications ISO7816-3 and with (or without) EMV scope.

ICC activation

cICC_ACTIVATE
cICC_ACTIVATE + <P>

When a card is present inside the reader, or a SAM is introduced into its socket, this command has to be used to power on, and to activate the ICC (chip card or SAM).

If this activation is successful, the reader returns the ATR frame from the newly activated chip.

The parameter <**P**> is optional; it permits to set the specification to be applied for the ICC activation

- <**P**> = 0: to activate only synchronous cards (memory card or I²C card)
- <**P**> = 1: to activate only asynchronous ISO card conformed to ISO 7816-3
- <**P**> = 2: to activate payment chip card, with respect of the standard EMV (Europay, Mastercard and Visa) specification. This parameter is mandatory to use the reader as an "EMV – Level 1" device.

When this parameter <**P**> is **not** given, the activation sequence is performed as following:

- Power ON the chip and start the asynchronous clock signal
- Decode the ATR send by the chip (if the chip is a micro-processor unit)
- If the ATR is valid, its value is send as response to the host. This ATR always starts with the character $3B_h$ or $3F_h$
- If the ATR is not valid (transmission error, or data outside specifications range) or if the chip does not respond, the chip is deactivated (powered off)
- If the ATR was not valid, the chip is activated once more. The clock line is activated in synchronous mode (50 kHz)
- Decode the data send by the chip (if the chip is a synchronous memory card)
- If this ATR data is valid, its value (which is always 4-bytes length) is returned as response to the host.
- If no data is returned by the chip, the terminal deactivates it
- If the chip returned no data, the chip is then activated for a third time. The reader uses the I2C protocol to get data from the card (if the chip is a I²C memory card)
- If the chip is responding at this request, the terminal returns an 8-bytes length frame. (Normally data read from address 0000_h to 0007_h).
- If there's still no valid response, the reader abort activation sequences, and the error code **rERROR** is transmitted to the host.

The activation is always first performed, by using the asynchronous protocol (ISO 7816)

If this first activation fails, then a second activation is performed, using the synchronous protocol (for memory cards)

If this second activation fails, then a third activation is performed, using the Philips I²C protocol.

If the mode “Auto ATR” is activated (in the reader’s configuration data), the chip activation procedure is automatically performed, as soon as a card is detected into the seated position. If the activation is successful, the ATR is automatically returned to the host, and the card is locked

The user has to take care of the returned ATR; he must set the appropriate memory card protocol (see command **cMEM_CARDTYPE**) for future synchronous data transfer

Note: The use of the optional parameter **<P>** with the command **cICC_ACTIVATE** is only defined on readers with firmware Version released after V2
So we recommend updating all readers with Version 2.0 (or higher)

ICC deactivation

cICC_DEACTIVATE

This command is used to power down the chip card, as defined into ISO 7816-3 and EMV.

Action:

- Deactivate the chip
- Clear ICC data buffer.
- Transmit a **rACK** as acknowledge

Note: If an activated card is about to be prematurely withdrawn, this command is automatically executed to guaranty the card deactivation ISO sequence.

ICC / SAM selector

cICC_SELECT + <N>

This command tells the reader which connector is to be used with succeeding chip commands. This command does not alter the link with the previous one.

<N> = ‘0’ (00_h or 30_h): to select the user chip card

<N> = ‘1’ (01_h or 31_h): to select the first SAM module

<N> = ‘1’ to ‘*N*’ to select one of the *N* available SAMs

Action:

- Set the current chip connector
- Transmit a **rACK** as acknowledge
- If <*N*> is out of range, the code **rUNAVAILABLE** is returned

Note: After a reader reset/restart, the default connector is always the user card one.

ICC data transfer

cICC_TRANSFER (or **cICC_READ**)

This request is used to pass an APDU command to an ICC (Asynchronous chip: the user chip card, or a secure module) and get the APDU response returned by the ICC.

APDU format are conformed to the structure defined in ISO 7816-4

The APDU command shall be included in the request:

cICC_TRANSFER CLA INS P1 P2 Lc …Data… Le

Where:

CLA, INS, P1, and P2 are mandatory

"Lc", "Data" and "Le" are optional.

Action:

- Clear ICC data buffer
- Check the given C-APDU syntax. If it is not correct, the error code **rINVALID** is returned. (No communication with the chip is performed)
- Check that the chip card is still active. If the card has been prematurely removed, or an electrically shutdown has occurred, the error code **rINVALID** is returned.
- Perform the data exchange with the chip card
- If successful, the returned data (if present) and the status word SW1 / SW2 are returned to the host, as R-APDU:

  …Response Data… SW1 SW2
- If unsuccessful, the card is automatically deactivated, and the error code **rERROR** is returned.

This command is independent of the card protocol: T=0 or T=1.

Complement information about ICC transfer

- The reader never performs any **PTS** (Protocol Type Selection) sequence.
- If the inserted card is conformed to protocol T=1, the reader automatically performs the SBlock[IFSD] sequence just before the first data exchange. (This sequence sets the reader buffer size to 254 bytes, as defined into EMV2000)

ICC data transfer (write only)

cICC_WRITE

This request is used to pass an APDU to a microprocessor card where only the ISO status word is expected as response.

Note: This command now acts exactly like `cICC_TRANSFER.`

It is obsolete, and is only present for USI compatibility. We suggest using the command **cICC_TRANSFER** for all data transfer with an asynchronous ICC

3.1.7. Memory card driver

This chapter defines the functions to be used to communication with synchronous card i.e. many kinds of memory cards.

Because there're many types of card, the user has to precise, whose communication protocol has to be used for

So the command **cMEM_CARDTYPE** has to be called prior to the command **cMEM_TRANSFER**.

Setting the current memory card protocol

cMEM_CARDTYPE + <Type> [+ <Page size>]

This command is two (or three) characters length.

It has to be used to set the type of protocol for the communication with the inserted memory card

The following values of **<Type>** are defined:

- '0': (00_h or 30_h): Siemens SLE 4406, SLE 4436, Thomson ST1305
- '1': (01_h or 31_h): Siemens SLE 4418, SLE 4428, Gemplus GPM8K
- '3': (03_h or 33_h): Siemens SLE 4432, SLE 4442
- '4': (04_h or 34_h): I²C protocol: with 1-byte address
- '5': (05_h or 35_h): I²C protocol: with 2-bytes address

The optional parameter <Page size> defines the memory size of one page from i²c cards.

The following default values are set:

- For type '4': Size = 08_h (for 8 bytes/page)
- For type '5': Size = 20_h (for 32 bytes/page)

Memory card Data transfer

cMEM_TRANSFER

To be conformed to ISO format (like with microprocessor card), requests for memory cards also use the APDU format

The APDU command shall be included in the request:

cMEM_TRANSFER CLA INS ADDR_H ADDR_L Lc … Data …

Or

cMEM_TRANSFER CLA INS ADDR_H ADDR_L Le

Or

cMEM_TRANSFER CLA INS ADDR_H ADDR_L Lc … Data … Le

With:

CLA is mandatory, but its value may always be null. This parameter is ignored.

INS defines the command to be performed. The defined values are

- **INS_READ_BINARY,** coded $B0_h$: to read, and get data bytes from the card. The number of data to be returned has to be set in **Le**. The reading will start from the address given by **ADDR.**
- **INS_WRITE_BINARY,** coded $D0_h$: to write data onto the card. The writing will start at the address given by **ADDR,** and **Lc** defines the length of the **Data** part.
- **INS_PASS_VERIFY,** coded 20_h: to unlock the card's write-protection. The length of the given password has to be set in **Lc**. The given password (or PIN code) will be set into the **Data** part.
- **INS_PASS_CHANGE,** coded 88_h: to modify the protection code. The length of the new password has to be set in **Lc**. The new password (or PIN code) will be set into the **Data** part. As response, the password, read back from the card, is returned.
  The user has to take care that the returned password is identical to the given one.
- **INS_RESTORE_DATA,** coded $C0_h$: to be used for pre-payment card (SLE4406) with units. This command is used to restore bits (units). It shall be used with one data byte. This byte will be written at the given address, to automatically erasing the 8 bits from the next counter (at the following address)
- **INS_SET_PROTECTION,** coded $C1_h$: to set a permanent write protection on some memory areas. To protect an area, it is necessary to input again the data. The protection flag will only be set if the input data and the card data are identical.
- **INS_READ_PROTECTION,** coded $C2_h$: to read the protection status from each memory area on the card. The byte **'s'** (73_h) is returned to indicate a protected area. A null byte (00_h) is returned for each address that is still able to be written

ADDR defines the address (first High byte, then Low byte), where readings or writings operations have to start.

Lc: Length of the DATA part

Data: Data to be written onto the card

Le: Length of the expected data the card has to return as response

Action:

- Check the given C-APDU syntax. If it is not correct, the error code **rINVALID** is returned. (no communication with the card is performed)
- Check that the card is still active. If the card has been prematurely removed, or an electrically shutdown has occurred, the error code **rINVALID** is returned.
- The reader interprets the request, then perform the data exchanges with the memory card, using the communication protocol set by the command **cMEM_CARDTYPE**
- If successful, the returned data given by the card, followed by the status word SW are returned to the host, as an R-APDU.
- If unsuccessful, the terminal only returns a status word; its value indicates the type of error.

The returned Status Word **SW** takes one of the following values:

- **SW_OK** coded 9000_h means the read/write process has been successfully performed.
- **SW_INS_ERROR** coded $6E00_h$: means that the given command cannot be performed, or is not compatible with the current selected card type.
- **SW_ADDR_ERROR**, coded $6E01_h$: means that the given address, where writing has to begin, is not allowed. This case may appears with some I²C cards, where data block writings may start at the beginning of the memory page.
- **SW_LC_ERROR**, coded $6E02_h$: means that the given parameter (DATA part) is not compatible with the associated command (INS).
  Example: using **INS_WRITE_BINARY** with no data, so **Lc** is null
- **SW_LE_ERROR**, coded $6E03_h$: means that the value **Le** isn't compatible with the associated command.
  Example: using **INS_READ_BINARY** with **Le** being null
- **SW_COMMAND_EXE**, coded $6E10_h$: means that a problem occurs during the command transmission to the card. Transmission is then aborted
- **SW_PROCESS_EXE**, coded $6E11_h$: means that a problem occurs during the processing phase, by the card. In this case, the read/write process is aborted.
- **SW_LOCKED**, coded $6E12_h$: means that the writing command is not allowed, because, the PIN code has not previously been successfully checked.
- **SW_EXE_ERRORS**, coded $64XX_h$: This error code is generally returned when the requested action has not been completely performed. The 2nd SW2 code (XX_h) contains the number of errors that has occurred during the process phase.
- **SW_PASSWD_ERRORS**, coded $65XX_h$: This error code is returned when the password (PIN) verification has failed. The 2nd byte SW2 contains the remaining PIN-trial count. When the value 6500_h is returned, it means that the card is definitely locked.

Examples:

I want to read 18 bytes from address 120_h to address 131_h. I should send:

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**$B0_h$**	**01_h**	**20_h**			**12_h**

The returned response could look like

Data (18 bytes)	SW
12 34 … 1A FF 1F 00	**9000_h**

I want to write the 5 bytes “12 34 56 78 AA” at the address 50_h to 54_h

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**$D0_h$**	**00_h**	**50_h**	**05_h**	**12 34 56 78 AA_h**	

When the process is done, the returned data is

Data	SW
	90_h 00_h

I input the PIN code “12_h 34_h”to unlock the card protection

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**20_h**	**00_h**	**00_h**	**02_h**	**12_h 34_h**	

I change the PIN code, I want to replace it with 43_h 21_h. I will get the

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**88_h**	**00_h**	**00_h**	**02_h**	**34_h 21_h**	**2**

The reader will send back the new PIN code, read from the card memory.

I want to write the 5 bytes “12 23 34 45 56” at the address 0160_h to 0164_h and then read back 16 bytes from address 0160_h to $016F_h$

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**$D0_h$**	**01_h**	**60_h**	**05_h**	**12 23 34 45 56_h**	**10_h**

Because Le is not null, the reader will automatically performed a “read” procedure after the “write” one. I.e. this command is similar to these successive ones

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**$D0_h$**	**01_h**	**60_h**	**05_h**	**12 23 34 45 56_h**	
00_h	**$B0_h$**	**01_h**	**60_h**			**10_h**

I want to get the write-protection status of the area addressed with 11_h to $1A_h$

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**$C2_h$**	**00_h**	**11_h**			**$0A_h$**

The returned response could look like

Data	SW
73_h 73_h 73_h 00_h 73_h 00_h 73_h 00_h 00_h 00_h	**9000_h**

Now I want to protect the area at the address 19h and 1Ah, that contains 12_h and 34_h

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**$C1_h$**	**00_h**	**19_h**	**02_h**	**12_h 34_h**	**02_h**

If successful, the returned response should look like

Data	SW
73_h 73_h	**9000_h**

I use a bit-counter card (SLE4406). I want to restore (to FF_h) the 8 bits of the address $0B_h$ by writing one bit (to zero) in the previous (MSB) counter address $0A_h$

CLA	INS	Add-h	Add-l	Lc	Data	Le
00_h	**$C0_h$**	**00_h**	**$0A_h$**	**01_h**	**$F7_h$**	**02_h**

If successful, the returned response should look like

Data from addresses $0A_h$ and $0B_h$	SW
$F7_h$ FF_h	**9000_h**

3.1.8. Retransmission

Previous response retransmission

cDEV_RETRANSMIT

This command is the request to ask for the reader's previous message retransmission, (for example if a transmission error occurs during the last response phase).

Action:

- The readers transmits the previous response again

3.1.9. Maintenance

The below functions are only available on the motorized models 855 / 869. It permits an easy checking of each reader component.

Motor command

cMAINTENANCE + 'M' + <P>

This command is only available on motorized model: 855 / 869. It has only to be used for some maintenance tests; it allows driving the motor for the card transport

Action:

<P> = '0' (00_h or 30_h): The motor stops

<P> = '1' (01_h or 31_h): The motor runs forward (like to intake a card)

<P> = '2' (02_h or 32_h): The motor runs backward (like to return the card back)

Shutter command

cMAINTENANCE + 'H' + <P>

This command allows driving the shutter.

Action:

<P> = '0' (00_h or 30_h): The shutter is closed

<P> = '1' (01_h or 31_h): The shutter is kept opened

Switches status

`cMAINTENANCE + 'S'`

This command asks of the status of the 5 (or less) status of a motorized reader. An ASCII character codifies each status

***** means "Card detected" **-** means No detection

! means "Shutter is closed" **=** means "Shutter is opened

The response is 6-character length message; the first char is space (20h) and is followed by the successive 5 status characters:

"Card before the shutter" – "Shutter position" – "Card behind the shutter" – "Card behind the roll" – "Card inside (or at CHIP position)

Ex: A card is inside the reader, the shutter is closed ➔ the last 2 status are activated

The reader returns: **- ! - * ***

Ex: A card is at front end, the shutter is opened:

The reader returns: *** = * - -**

Magnetic card **D**ecoding test

`cMAINTENANCE + 'D' + <P>`

This command makes the card move under the magnetic head.

Action:

<P> = '1' (01_h or 31_h): Card is moved inside the reader => Decoding in forward direction

<P> = '2' (02_h or 32_h): Card is moved outside the reader => Decoding in backward direction

The response of the decoding is a 3-char ASCII message. Each character gives the result of each track

'**1**' is returned when the track has been successfully decoded (standard ISO mode)

'**0**' is returned when the track has not been decoded (or with errors)

'-' is returned when the track is unavailable (in case of use of 1 or 2-tracks magnetic head)

Ex: All 3 tracks have decoded without error

The readers returns: **1 1 1**

Ex: Only tracks 1 and 2 are correct; but an error occurs reading track 3

The readers returns: **1 1 0**

Ex: Track 1 is unavailable, an error and track 2, and track 3 is OK

The readers returns: **- 0 1**

2. Responses from the reader

Reader start signal

rStart

This message is the first one sent by the reader, when it starts and gets ready to process new command coming from the host.

This message is sent too, when the reader has performed a warm reset, after having received the command **cRESET** from the host.

Positive acknowledge

rACK

This response is sent to acknowledge requested actions that have been correctly executed.

Negative execution

rERROR

When the requested command could not be correctly executed, because a process error has been detected, this response is returned

No magnetic data

rNO_DATA

When the request is a magnetic data reading, this response is returned when no data has been decoded from the requested track.

No magnetic card

rNO_MAG_CARD

When the request is a magnetic data reading, this response is returned when the reader is in the arm mode, and no magnetic stripe has been detected during the card insertion and/or removal.

Magnetic stripe detection signals

rMAG_DETECT_ON and **rMAG_DETECT_OFF**

These both messages are sent to the host when the command **cARM_DEBUG** is activated, and the reader is just about to detect the beginning and the end of the magnetic stripe signal.

Invalid command

rINVALID

This response is returned when the received command is unknown or undefined, or if the requested order cannot be performed in the current process configuration state.

Corrupted message

`rPROT_ERROR`

This response is returned when the reader receives a corrupted message, with a protocol error (i.e. the checksum is not correct).

Unavailable order

`rUNAVAILABLE`

This response is returned when the hardware is not available to complete the request

3. Codes definition

This chapter enumerates the *ascii* and *hexa* codes of all previous defined commands

3.3.1. Commands send by the host

cABORT	[ESC]	1B
cARM	‘P’	50
cARM_DEBUG	‘p’	70
cCARD_CAPTURE	‘|‘	7C
cCARD_LOCK	‘{‘	7B
cCARD_UNLOCK	‘}’	7D
cCARD_POSITION	‘8’	38
cDEV_DESCRIPTORS	‘7’	37
or	‘9’	39
cDEV_RESET		7F
cDEV_RETRANSMIT	‘%’	25
cDEV_STATUS	‘$’	24
cICC_ACTIVATE	‘N’	4E
cICC_DEACTIVATE	‘n’	6E
cICC_READ	‘a’	61
cICC_SELECT	‘C’	43
cICC_TRANSFER	‘a’	61
or	‘A’	41
cICC_WRITE	‘A’	41
cLED_GREEN_ON	‘L’	4C
cLED_GREEN_OFF	‘l’	6C
cLED_GREEN_FLASHING	‘(’	28
cLED_RED_ON	‘M’	4D
cLED_RED_OFF	‘m’	6D
cLED_RED_FLASHING	‘)’	29
cMAG_ISO_T1	‘Q’	51
cMAG_ISO_T2	‘R’	52
cMAG_ISO_T3	‘S’	53
cMAG_FMT_T1	‘q’	71
cMAG_FMT_T2	‘r’	72
cMAG_FMT_T3	‘s’	73
cMAG_CUSTOM_T1	‘U’	55
cMAG_CUSTOM_T2	‘V’	56
cMAG_CUSTOM_T3	‘W’	57
cMAG_ERROR	‘I’	49
cMEM_CARDTYPE	`T`	54
cMEM_TRANSFER	‘B’	42
cMAINTENANCE	‘!’	21

3.3.2. Responses send by the reader

rACK	‘^’	5E
rERROR	‘*’	2A
rINVALID	‘!’	21
rMAG_DETECT_ON	‘(‘	28
rMAG_DETECT_OFF	‘)’	29
rNO_DATA	‘+’	2B
rNO_MAG_CARD	‘>’	3E
rPROT_ERROR	‘?’	3F
rSTART	‘:’	3A
rUNAVAILABLE	‘~’	7E

IV. Configuration & Download

The system environment is based around a microprocessor from the "H8 tiny" family from Hitachi.

This processor includes flash memory, which is divided into few independent segments.

- The "boot" programs
- The "serial number"
- The Card handler's "Application software"
- Its configuration data

The "boot" part is programmed, in factory, and might not be modified by the user. This program has to manage the downloading of new reader's applications firmware, and configuration data.

The "Application firmware" is the program that drives the reader and the communication protocol with the host system. This code may change over time, and must be easy to upgrade. This chapter explains the way to execute a new firmware downloading.

The configuration data has to be written with the application firmware. It contains some hardware settings or some program parameters.

1. The Boot program

This program is always the first one that is executed, when the reader starts.

When the reader starts running:

- It first set the serial communication parameters:
  38400 baud, even parity, 8 data bits, 1 stop bit
- It sends the character **`SIG_START`** ($3F_h$)
- It waits, during maxi one second, for a host request (i.e. **`SIG_VERSION`**)
- If the reader received a request during this time interval, it will return the response, and stay in the boot mode; the application program does **not** start. The reader is then ready to perform downloading processes; it just waits for boot commands.
- If the reader do not received a command during this one-second time interval, it checks the presence of an application firmware; and jump into it to start the application program.
- If no application program is loaded, the reader automatically resets after the one-second timeout. The reader will restart following the steps defined just above.

2. The “boot” protocol

4.2.1. Frame structure

In boot mode, the serial parameter setting is always:

Format: 1 start bit, 8 data bits, **Even** parity

Rate: 38400 baud

No flow chart

To perform flash memory readings, and new application down-loadings, the user has to send commands and data, conformed to the following protocol. A request or data frame, sent by the host to the reader, is always conformed to:

ESC	FCT	ADDR	[… DATA …]	CS

With:

ESC = $1B_h$

FCT:

Function code: an ASCII character (noted here with **FCT_*XXXXX***)

ADDR:

This field is mandatory. it’s the address where the data have to read, write, or erase.

With functions which do not need this data, a null value will be set.

The address is 2-bytes length. The MSB is send first.

DATA:

This is the Data to be programmed into the flash memory.

This field is only present when the function code is “**FCT_PROGRAM**”

This field is always absent for any other function codes.

Into a programming command frame, this field must always be 128 bytes length. Used data have to be set to the value FF_h

CS:

Is the “checksum”. An Exclusive OR (XOR) with all the bytes FCT, ADDR, and DATA (ESC not included) computes this value.

When the reader device, running its boot module, receives a correct frame, it immediately sends back a acknowledge:

- ACK (06_h) to inform that the command frame syntax is correct. The reader can start the execution

or

- NAK (15_h) when the frame is corrupted (because of a bad checksum)

Following the ACK byte, the response data (if they are) are returned. If the request is unknown or not allowed to be executed, the error code **SIG_NOT_ALLOWED** will be returned.

4.2.2. The commands

All commands, allowed by the used to perform uploading or downloading operations, are described here above:

Boot firmware version

FCT_VERSION – 0000_h

This request returns an ASCII string that describes about the version of the boot module. In fact, the boot program is divided in two modules:

When the "loader" module is not active (after a reset-restart), the returned string will look like: "H8-Boot V2.x"

When the "loader" module is active (after executing the command **FCT_DOWNLOAD**), the returned string looks like: "H8-Loader V2.x".

Read back data from flash memory

FCT_READ - <Addr>

It returns a set of 128 bytes, read from the flash memory, starting from the address **<Addr>** to **<Addr+127>**

All the flash memory area, from 0000_h to $7FFF_h$, is readable.

Loader activation

FCT_DOWNLOAD – 0000_h

This command activates the "Loader" module, to allow erase/writing operations to the flash memory.

If the "Loader" module is already active, this command is ignored

Erasing the flash memory

FCT_ERASE - <Addr>

Before writing new data, the flash area must be erased, i.e. all data must have the value FF_h. So Erasing must first be run before executing any writing process.

The flash memory is divided in few segments. One **FCT_ERASE** command erases all data from one segment.

The segment corresponding to
<Addr> = 1000_h
should be erased, just before performing a new application firmware downloading.

The segment corresponding to
<Addr> = $0C00_h$
should be erased, just before performing a new configuration downloading.

You should never use this command with other address values

During the erasing process, the reader returns some character '**E**' or 'B'. When the erasing process is done, the character **SIG_OK** is returned if the area has been successfully erased. The response **SIG_ERROR** could be returned, if the erasing process terminates with an error. This command has to be repeated (until 10 times), to perform a new erasing.

Note: This function is only allowed when the "Loader" module is active.

Verify memories blanking

FCT_BLANK - <Addr>

The segments corresponding to
<Addr> = $0C00_h$ or <Addr> = 1000_h

has to be checked with this command. If the corresponding flash area is correctly cleared, i.e. all the data from the segment are equal to FF_h, then the code **SIG_OK** is returned. This area (from **$0C00_h$** up to **$0FFF_h$**, or from **1000_h** up to **$7FFF_h$**) is then ready to be programmed again

If not, the error code **SIG_ERROR** is returned. The function **FCT_ERASE** has to be performed once more.

Writing data into the flash memory

FCT_PROGRAM - <Addr> - <128 data bytes>

This command has to be executed, many times, to write a new application or configuration data, to the flash memory.

Programming must always be done to an empty area. The flash memory is programmed 128 bytes at a time. If fewer then 128 bytes have to be written, values FF_h must be written to complete the data block.

<Addr> is set at **$0C00_h$** to download the configuration block.

<Addr> is set from **1000_h** to **$7F80_h$**, (with incrementing step = **80_h**) during the application firmware downloading.

During the writing process, the reader returns many times the character 'P'. When the writing process is done, the response **SIG_OK** is returned if the current area has been successfully written. If the programming process fails, the reader returns **SIG_ERROR**

Important: You should never use this command with other address values

Note: This function is only allowed when the "Loader" module is active.

Exit the boot mode

FCT_EXIT – 0000_h

This command makes the reader exit the boot mode, and restart. If the new firmware has been successfully downloaded, the reader should be able to start with the new application.

4.2.3. Boot codes definition

Commands send to the reader

FCT_DOWNLOAD	**'D'**
FCT_ERASE	**'E'**
FCT_BLANK	**'B'**
FCT_PROGRAM	**'P'**
FCT_READ	**'R'**
FCT_VERSION	**'V'**
FCT_EXIT	**'X'**

Responses from the reader

ACK	**06_h**
NAK	**15_h**
SIG_START	**'?'**
SIG_OK	**'1'**
SIG_ERROR	**'0'**
SIG_NOT_ALLOWED	**'n'**
SIG_PROGRAMMING	**‘P’**
SIG_ERASING	**‘E’**

3. The reader's configuration data

The reader's configuration data consists of some 64 parameters that the reader's application takes care to define how it has to work.

These data are written and saved into the flash memory, at the addresses **0C00$_h$** to **0C40$_h$**. The following table gives the significance of each byte. Default parameter values are **FF$_h$**

Offset	Meanings
0	= **00$_h$**: Magnetic Track 1 is not decoded = **01$_h$** to **FF$_h$**: Magnetic Track 1 is decoded
1	= **00$_h$**: Magnetic Track 2 is not decoded = **01$_h$** to **FF$_h$**: Magnetic Track 2 is decoded
2	= **00$_h$**: Magnetic Track 3 is not decoded = **01$_h$** to **FF$_h$**: Magnetic Track 3 is decoded
3	= **00$_h$**: No locking system present on the reader = **01$_h$** to **FF$_h$**: Reader equipped with a locking system
4	= **00$_h$**: No added capacitor connected = **01$_h$** to **FF$_h$**: Unlock at power fail detection (external capacitor)
5	= **00$_h$**: No chip card connector = **01$_h$** to **FF$_h$**: Chip card connector present
6	= **00$_h$:** No SAM connector on the reader's electronic board = **01$_h$** to **07$_h$**: Quantity of SAM connectors on the reader's board
7	= **00$_h$** or **FF$_h$**: No external SAMs board connected = **01$_h$** to **08$_h$ =** *'N'*: Extended SAM board with "*N*" connectors
8	Protocol USI2: Network Address = **00$_h$** or **FF$_h$**: No defined address: a null value will be used = **01$_h$** to **7F$_h$:** Defined network address
9	V24 Transmission Baud rate = **00$_h$** or **FF$_h$**: 38400 b/s (default rate) = **01$_h$:** 19200 b/s = **03$_h$:** 4800 b/s = **02$_h$:** 9600 b/s = **04$_h$:** 2400 b/s
10	V24 Transmission format = **00$_h$** or **FF$_h$**: No parity = **01$_h$:** Odd parity = **02$_h$:** Even parity
11	V24 Transmission flow control (reader's output signal) = **00$_h$** or **FF$_h$**: DTR is always active when the reader runs = **01$_h$:** DTR is only active when the reader is ready to get a frame
12	V24 Transmission flow control (reader's input signal) = **00$_h$** or **FF$_h$**: CTS signal is ignored = **01$_h$:** The reader waits for CTS to be active before transmitting
13	= **FF$_h$**: Reserved for future use
14	= **FF$_h$**: Reserved for future use
15	= **00$_h$** or **FF$_h$**: Reader without shutter = **01$_h$**: a shutter is connected to the read = **02$_h$**: an additional magnetic head equips the shutter
…	

16	= **00h** or **FFh**: No automatic Switch report mode = **01h**: Option "***Auto-Switch report***" is activated
17	= **00h** or **FFh**: No automatic chip card activation = **01h**: Option "***Auto ATR*** mode is activated : Inserted ICC is automatically activated
18	= **00h** : Option "***Arm-And-Ack***" is deactivated = **01h** to **FFh**: Option "***Arm-And-Ack***" is activated
19 to 63	= **FFh**: Reserved for future use

The options

"***Auto-Switch report***"

This option makes the reader automatically send a message, when a card detection status is changing. This message is similar to the response returned by the use of the command **cCARD_POSITION:**

"***Auto-ATR***"

This option makes the reader automatically activate the chip card, when a card is inserted, without having to send the command **cICC_ACTIVATE**

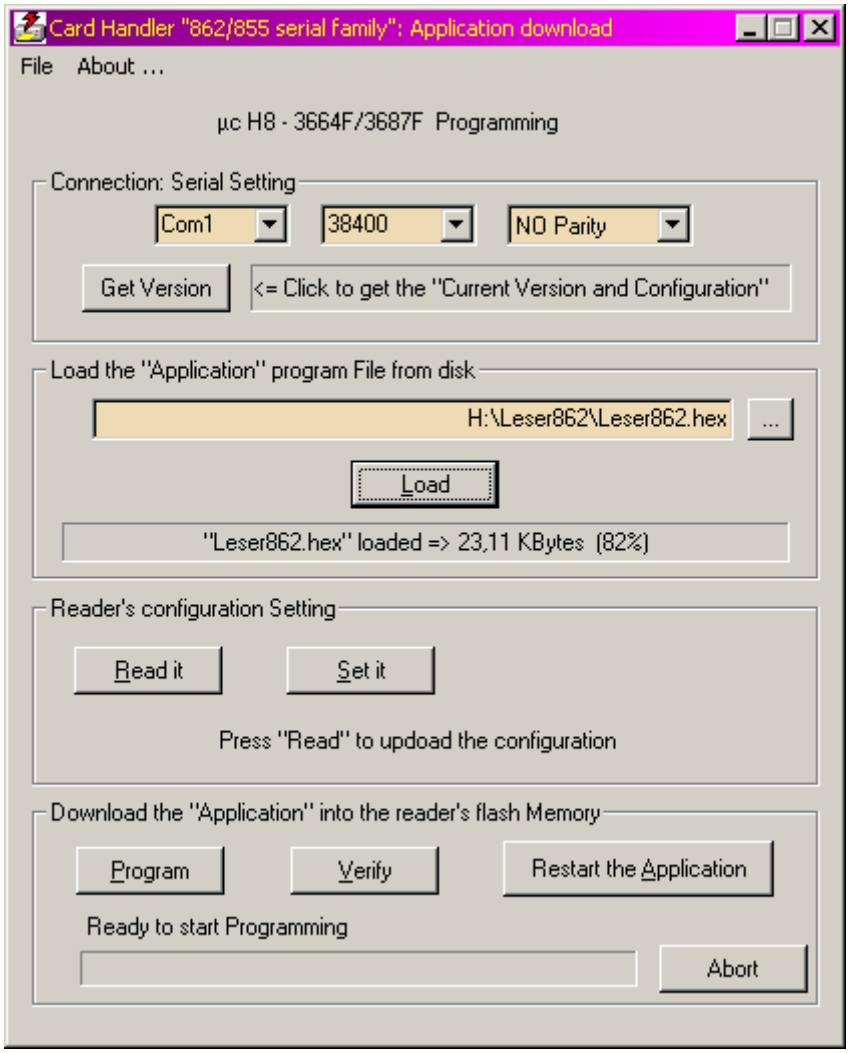
So when a card is inserted, its ATR is automatically transmitted to the host

"***Arm-And-Ack***"

After having sent the command **cARm**, this option makes the reader automatically transmit an **rACK** or an **rNO_DATA** when the reader exits the ARM mode, i.e. when the reader has (tried to) decoded a magnetic strip card. By default, this option is activated

Configuration tool: the program H8Prog862

To change or set the reader configuration, a Windows ™ tools, called **H8Prog862** is available.



This program could be downloaded from

http://www.hopt-schuler.com

or

https://www.ines-communication.com/svn/ddm/

or, you can also demand for this tool, by e-mail to info@ines-communication.com, we will transmit it.