Hybrid card controller

“862 / 867 / 868 / 855”

Serial & USB models

DLL Driver
Reference manual

Communication library V1.4

Dec. 2004

ddm hopt+schuler
Heerstraße 44
D-78626 Rottweil
+49 741 / 26 07-0
+49 741 / 1 33 98
www.hopt-schuler.com

History

Date	Rev	Notes
2003-07-08	1.01	Initial draft
2003-10-22	1.02	First Release version
2004-08-30	1.03	Additional functions: USB reader + motorized reader
2004-12-15	1.04	Chip: Activation mode extension

Written by:

INES Communication
8, rue Claude Chappe
F 57 070 METZ

✆ +(33) 3 87 39 08 00
🖷 +(33) 3 87 39 08 04
✉ info@ines-communication.com

Author: Marc METZINGER

Table of contents:

1 System requirements

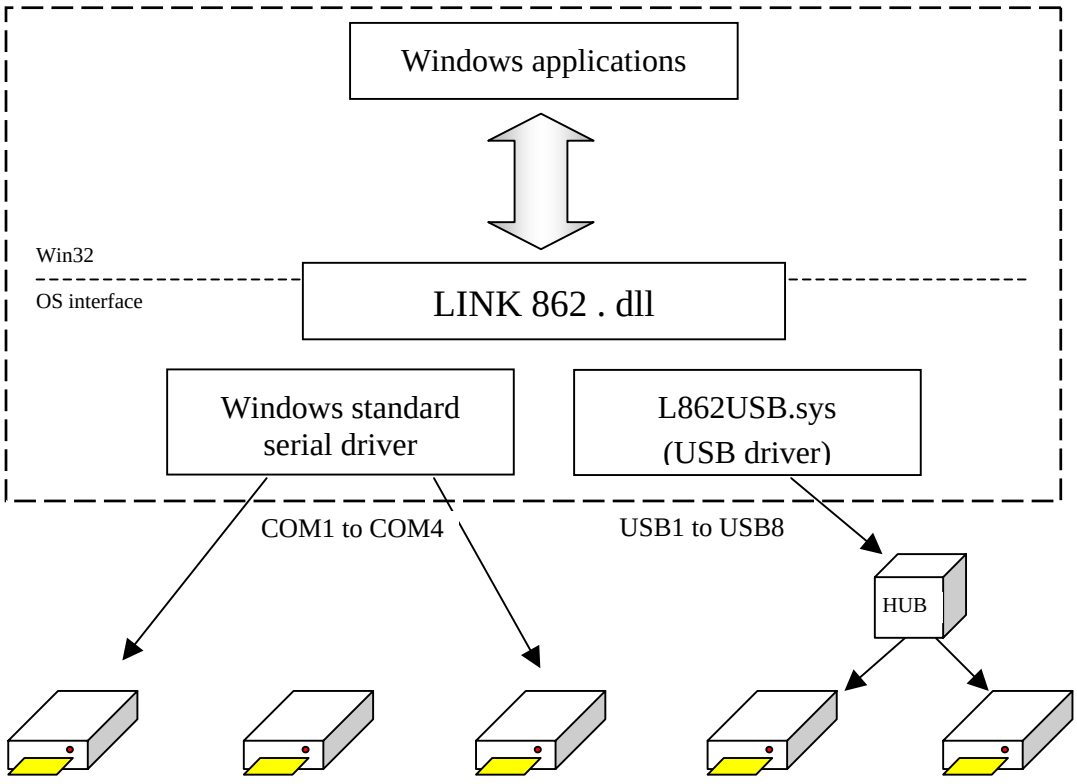
This driver is written to be installed on Microsoft operating systems:

- Windows 98 / ME
- Windows NT / 2000 / XP

Its form is a DLL file that has to be copied into the system directory, usually

- C:\Windows
- C:\WinNT

This library uses the standard serial driver (COM1 to COM4) for communication with serial reader. An additional USB Windows driver needs to be install, to allow communication with USB reader.



1.1 At runtime:

To execute the final application, only the following files are needed

- Link862.dll
- Mfc42.dll (included into last versions of Windows ™)

When an USB reader is connected, the files

- L862usb.inf
- L862usb.sys

have to given to allow the Windows installation of the USB driver.

1.2 During development

To develop applications with Microsoft Visual C++ V6.0, you need

- The header file “Link862.h”
- The library file “Link862.lib”

to be included into the project.

To develop applications with Microsoft Visual Basic, you need

- The module file “Link862.bas”

to be included into the project.

Of course, other languages or development kits can be used. The library function entries have been defined to be conformed to Microsoft Windows API function calls.

The communication rule is conformed to Master-Slave principle. The controller unit first sends a request frame, and then the card reader answers with the response frame. The reader never starts any communication on one's own authority.

1.3 Other document

This document only describes some functions that may used to easily develop some Windows™ applications. It must be completed with the document

“Hybrid Card Controller 862 / 867 / 868 / 855
Command reference manual
Protocol description”

to understand and get all the information about how the “ddm hopt+schuler 862” reader runs

2 Protocol descriptions

This chapter describes, in summary, the protocol specifications

For more details, refers to documents “862 – Command reference manual”

2.1 RS232 Transmission data format

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.

The reader always sends a response, when it receives a request with a null address.

But when the reader receives a request with a no-null address, the reader checks it, and compares it with its own one.

The reader’s address has to be set in the reader’s configuration data block

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 zero, the data part is generally encoded in ASCII mode. The control character ‘EOT’ 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. Most commands or responses are only 1-char length. See the document “862 – Command reference manual” to know more about the data. Nevertheless, the next chapter defines the functions that will construct the data frames.

BCC:

Is the “block check character”. Its value is computed by exclusive OR of all preceding bytes (SOH and EOT bytes are included).

3 The functions: reference guide

3.1 Rules

Driver setting functions return a Boolean code. This code is TRUE when the function has been successfully processed.

Most of the communication functions return short-signed integer (coded with 2 bytes) as error code. When the function fails, a null or negative code is returned. Different values are defined to give the reason of the error:

- When the function returns L862_OK (= 1):
  The reader has returned the positive acknowledge

usiRet_ACK	or	The reader has returned the requested data

- When the function returns L862_NEGATIVE (= 0)
  The reader has returned a negative acknowledge message like

usiRet_ERROR	usiRet_NO_DATA
usiRet_INVALID	usiRet_NO_MAG_CARD
usiRet_PROT_ERROR	usiRet_HARD_UNAVAILABLE

- When the function returns an negative error code L862_XXX, it means the function has not been successfully performed. The error code has to help the application developer to find the reason of this undesired result.

usiRet _NO_RESPONSE	usiRet _ERR_ARGUMENTS
usiRet _OVERRUN	usiRet _ERR_PORT_CLOSED
usiRet _BUFFER__SMALL	…

3.2 Driver installation

The following functions have only to be called one time, during the application process. They consist of system configuration, and the communication port managing

L862_Install

Syntax:

C, C++: BOOL **L862_Install** (HWND hWnd, void * EventFct, int WM_862);

V.Basic: Function **L862_Install** (ByVal hWnd As Long, ByVal EventFct As Long, ByVal WM_862 as Long) As Boolean

Purpose:

This function is the first one to be called, once only, at the beginning of the user application. It consists to override the main application Windows function, and install the receiving event process from future "L862" channels that will be opened with L862_Open function.

At the end of the application, just before exit, the function **L862_Free** has to be processed

Return value:

If the driver L862 has been correctly installed into the Windows systems for this application, the function returns the value TRUE.

If this driver could not been installed properly, the function will return FALSE

Parameters:

HWnd:

The handle of the mainframe window of the user application. When a reception event occurs, the driver sends the Windows message WM_862 to the user application, and then gives the possibility to call the user "Event function"

EventFct:

The address of the user event function. Into its application, the user can define a callback function that will be executed to get data response, coming from the reader.

This event function shall have the following structure:

C / C++: void __stdcall MyEventFct (short hPort, short RetCode);

V. Basic: Sub MyEventFct (ByVal hPort, ByVal RetCode as Integer)

- hPort is the handle of the link, which returns the response event
- RetCode is the first data byte of the frame transmitted by the reader. If this character is one of the known returned code (usiRet_ACK, usiRet_NO_DATA, usiRet_POS_*xxxx* …) the application can immediately take it as an event from the reader.
  If this code value is undefined, the application has to call the "GetResponse" function, to capture the frame the reader had sent, and to analyze it.

A null value (null pointer) can be given is no user event catching function has to be defined.

If defined, this function will be called, each time a message is coming from the reader (outside the execution of high-level functions, which include responses catching)

WM_862:

This value set the “Windows message” code this driver will use. The user has to precise a value that is not used into its application to avoid undesirable effects. In all cases, this value must be higher than WM_USER (because lower values are reserved for the Windows O.S. uses).

If this parameter is set to 0, the function will set it to a default value (equal to WM_USER+99)

L862_Free

Syntax:

C, C++: bool **L862_Free** (void);

V.Basic: Function **L862_Free** () As Boolean

Purpose:

When all link862 ports have been closed, this function has to be called to uninstall the Windows events catching systems, and it makes the Windows system to return to its original mode.

Return value:

If the driver L862 has been correctly uninstalled, the function will return the value TRUE.

If an abnormality happens, the function returns FALSE

Get862_DllVersion

Syntax:

C, C++: const char * **Get862_DllVersion** (void);

V.Basic: Function **Get862_DllVersionB** (ByVal strBuffer as String,
ByVal BuffLen as Integer) As Integer

Purpose:

This function only returns an ASCII string, informing about this driver library version.

Return value:

The C version returns the pointer onto the requested string

The Basic version copies it into a given buffer, and returns the length of this string

L862_Open

Syntax:

C, C++: short **L862_Open** (const char * strPort);

short **L862_OpenSerial** (const char * strPort, int Baudrate, char Parity);

short **L862_OpenUsb** (short iPort);

V.Basic: Function **L862_Open** (ByVal strPort as String) As Integer

Function **L862_OpenSerial** (ByVal strPort as String, ByVal Baudrate as Long, Byval Parity As Byte) As Integer

Function **L862_OpenUsb** (ByVal iPort as Integer) As Integer

Purpose:

This function is to use to open a new communication channel, and establish a link with '862' or '867'-card controller devices.

Note: With the function "L862_Open", and a COM port, the default format is set:
38400-baud – no parity – 8 data bits – 1 stop bit

Return value:

If the channel is successfully opened, it returns a handle (positive value) to identify this channel. This handle is to be used in all others L862 functions.

If it fails, a negative value as error code is returned.

Parameters:

strPort:

The name of the port channel to use (for example "COM1", or "USB")

iPort (with "L862_OpenUsb"):

The index of the USB port to open. If this value is null, the system automatically looks for an available USB port.

Baudrate (with "L862_OpenSerial"):

The value of the baud rate to be set (for example '19200'). This value has to be conformed to the reader settings

Parity (with "L862_OpenSerial"):
'N' or '0' or 00_h : No parity
'O' or '1' or 01_h: Odd parity
'E' or '2' or 02_h : Even parity

L862_Close

Syntax:

C, C++: void **L862_Close** (short hPort);

V.Basic: Sub **L862_Close** (ByVal Port as Integer)

Purpose:

This function is used to close the communication channel, and frees this port for other uses. If the given channel is opened, it will be closed; all other cases will be ignored

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

3.3 "Low level" accesses

The low-level functions are the basic ways to send order or request frames, and to get the responses.

L862_SendCommand

Syntax:

C, C++: short **L862_SendCommand** (short hPort, short Cmd);

short **L862_SendCommandEx** (short hPort, short Cmd, const char * Param, short ParamLen);

V.Basic: Function **L862_SendCommand** (ByVal hPort As Integer , ByVal Cmd As Integer) As Integer

Function **L862_SendCommandEx** (ByVal hPort As Integer , ByVal Cmd As Integer , ByVal Param As String , ByVal Len As Integer) As Integer

Purpose:

These functions are used to only send a command frame to the reader. This function does not wait for the reader's response.

We don't explain here the content of each type of request. Please refer to "862 – Command reference manual" to get all details about the syntax of a command and its parameters

Return value:

If the command frame has been successfully sent, this function returns a positive code (i.e. L862_OK)

If the function call fails, a negative value is returned.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Cmd:

This ASCII character identifies the command code, that has to be sent, to perform the request. The following values are defined:

usiCmd_ARM	usiCmd_ICC_ACTIVATE
usiCmd_ARM_DEBUG	usiCmd_ICC_DEACTIVATE
usiCmd_ABORT	usiCmd_ICC_WRITE
usiCmd_CARD_LOCK'	usiCmd_ICC_READ
usiCmd_CARD_UNLOCK	usiCmd_ICC_SELECT
usiCmd_DESCRIPTION	usiCmd_ICC_WRITE
usiCmd_LED_*color*_ON	usiCmd_MAG_ISO_*Ti*
usiCmd_LED_*color*_OFF	usiCmd_MAG_FMT_*Ti*
usiCmd_LED_*color*_FLASH	usiCmd_MAG_CUSTOM_*Ti*
usiCmd_POSITION	usiCmd_MAG_ERROR
usiCmd_RESET	usiCmd_VERSION
usiCmd_RETRANSMIT	
with *color* = RED or GREEN	with *Ti* = T1 or T2 or T3

Param:

Pointer on the additional parameter data

Len:

Size (in bytes) of the parameter data buffer “Param”

Most of the commands are only 1-byte length: the command code. So the function L862_SendCommand may be used

Ex 1: To arm the reader:

```
rc = L862_SendCommand ( hPort, cARM );
```

The function L862_SendCommandEx has to be used when the command frame is more than 2-bytes length.

Ex 2: To send an Apdu to a chip card:

```
rc = L862_SendCommandEx ( hPort, cICC_WRITE,
      My_ApduCmd, My_ApduLength );
```

L862_GetResponse

Syntax:

C, C++: short **L862_GetResponse** (short hPort, short *Len, char * pData,
long Timeout);

V.Basic: function **L862_GetResponse** (ByVal hPort As Integer, ByRef Len As Integer,
ByVal pData As String, ByVal Timeout As Long) As Integer

Purpose:

This function has to be called to get the response frame from the reader.

This function only returns, when the response is received from the reader, or if the timeout has expired. But during this waiting time, the Windows OS messages (like WM_TIMER or WM_COMMAND) continue to be processed. This waiting time can be aborted by calling the function L862_Abort

Return value:

The value L862_OK (= 1) is returned when a valid response frame has been received.

The value L862_NEGATIVE (= 0) is returned when a negative acknowledge code has been received:

usiRet_ERROR	usiRet_NO_DATA
usiRet_INVALID	usiRet_NO_MAG_CARD
usiRet_PROT_ERROR	usiRet_HARD_UNAVAILABLE

All other cases are abnormal situations. Returned data may be incomplete, or misses. So the returned value (when the "L862_GetResponse" function fails,) is a negative error code.

L862_NO_RES	L862_ERR_ARGUMENTS
L862_OVERRUN	L862_ERR_PORT_CLOSED
L862_BUFFER_SMALL	…

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Len:

Double pointer:

- At function call, its value has to be set with the given "pData" buffer size.
- When function returns, this pointer will be set with the returned DATA length.

If the given buffer is too small to store all the response data, the function returns with the error code L862_BUFFER_SMALL, and only the beginning part of the response are returned.

The value NULL can be enter, if no data bytes are expected.

pData:

Pointer to storage buffer to received the returned data

The value NULL can be enter, if no data bytes are expected.

Timeout:

It's the maximum timing, in milliseconds, the function has to wait for the response.

A negative value (TIMEOUT_INFINITE) can be set to wait infinitely at any response to be returned. Nevertheless, this option is not recommended.

Get862_ReturnedCode

Syntax:

C, C++: short **Get862_ReturnedCode** (short hPort);

V.Basic: Function **Get862_ReturnedCode** (ByVal Port as Integer) As Integer

Purpose:

This function may be used when the **L862_GetResponse**, or any high-level function, returns the code L862_NEGATIVE. This situation means the reader has returned an error code (usiRet_ERROR, usiRet_INVALID, usiRet_*xxxxx* …).

This function allows getting this error code, transmitted by the reader

Return value:

The value 'UsiRet_xxxx'

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Get862_ReturnedText

Syntax:

C, C++: const char **Get862_ReturnedText** (short Code);

V.Basic: Function **Get862_ReturnedTextB** (ByVal Code as Integer, ByVal Text As String, ByVal Len As Integer) As Integer

Purpose:

This function returns an ASCII text that describes the given error, returned by the function Get862_ReturnedCode.

Parameters:

Code: value of usiRet_*XXXX*

Text: Buffer where the requested string will be written

Len: Size of the 'Text' buffer

Get862_ErrorText

Syntax:

C, C++: const char **Get862_ErrorText** (short FctCode);

V.Basic: Function **Get862_ErrorTextB** (ByVal FctCode as Integer, ByVal Text As String, ByVal Len As Integer) As Integer

Purpose:

This function returns an ASCII text that describes the 'returned code', returned by the communication functions L862_GetResponse, L862_ExecXxxx, L862_ReadXxxx …

Parameters:

FctCode: Code returned by most of these library's functions

Text: Buffer where the requested string will be written

Len: Size of the given 'Text' buffer

L862_IsRunning

Syntax:

C, C++: BOOL **L862_IsRunning** (short hPort);

V.Basic: Function **L862_IsRunning** (ByVal Port as Integer) as Boolean

Purpose:

This function has to be used to just to know if a response waiting process is in run.

Return value:

If the function "L862_GetResponse" or a high-level 862-function is currently running in another thread or event function, the value TRUE is returned.

Else, the value FALSE is returned.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Exemple:

For example, this function may be used in a WM_CLOSE handling function. If it returns the value, you may not continue to close the application.

L862_Abort

Syntax:

C, C++: BOOL **L862_Abort** (short hPort);

V.Basic: Function **L862_Abort** (ByVal Port as Integer) as Boolean

Purpose:

This function has to be used to prematurely break the execution of the function call "L862_GetResponse".

Return value:

If a function "L862_GetResponse" (or any of the high-level 862-functions) was active, the value TRUE is returned.

If no "L862_GetResponse" had been activated, this function call has no effect, and the value FALSE is returned.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

3.4 "High Level" functions

To simplify the use of data exchange functions, this set of following "high level" functions proposes an easier way to send order and to get formatted data from the reader.

The Arguments given at function call, are the formatted parameters to send, and the formatted data to be returned by the reader.

Most of these functions are only a combination of the two "low-level" functions:

- L862_SendCommand (list of parameters)
- L862_GetResponse (pointers on returned data + predefined waiting timing).

For these functions, which have generally very short process timing, the logic is to wait data response, just after having sent the request. In this way, you will find into the arguments:

- the list of parameter to send
- the list of the data to be returned

L862_ExecReset

Syntax:

C, C++: short **L862_ExecReset** (short hPort);

V.Basic: function **L862_ExecReset** (ByVal hPort As Integer) As Integer

Purpose:

This function is used to make the reader reboot, and restart again. So the reader is set in its default mode

When the reader starts (after being powered, or after a reset), it automatically sends a frame with Data = usiRet_START

Return value:

The positive code L862_OK is returned, after the reader has restarted and sent the "restart" message.

A null (L862_NEGATIVE), or negative value (L862_*ERROR*) is returned, if this function fails for any others reasons.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

L862_ExecArmToRead

Syntax:

C, C++: short **L862_ExecArmToRead** (short hPort);

short **L862_ExecArmToRead_Debug** (short hPort);

V.Basic: function **L862_ ExecArmToRead** (ByVal hPort As Integer) As Integer

Purpose:

Arming the reader means the reader clear its magnetic tracks data buffer, and becomes ready to decode new data for a new card pass under its magnetic head. This means the reader enters into the “arm” mode.

The standard version of this function only consists to sent the command “c862_ARM”, and to wait at the response “usiRet_ACK” to be returned by the reader.

The debug version of this function consists to sent the command “c862_ARM”, then to wait at the response “usiRet_ACK”. When the reader starts to detected a magnet stripe, the code usiRet_MAG_DETECT_ON is sent. When the magnetic decoding has been performed, the reader returns usiRet_MAG_DETECT_OFF. If no data has been decoded during the head pass, the code usiRet_NO_MAG_DATA is returned.

Return value:

The positive code L862_OK is returned, when the reader returns an acknowledge code c862_ACK.

A null (L862_NEGATIVE), or negative value (L862_*ERROR*) is returned, if this function fails for any others reasons.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

L862_ExecAbort

Syntax:

C, C++: short **L862_ExecAbort** (short hPort);

V.Basic: function **L862_ ExecAbort** (ByVal hPort As Integer) As Integer

Purpose:

This function has to be run to make the reader leave the “armed” mode.

When the “armed” mode is deactivated, the reader will not decode any data during a card head pass.

Return value:

The positive code L862_OK is returned, when the reader returns an acknowledge code usiCmd862_ACK.

A negative value is returned, if this function fails for any others reasons.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

L862_ExecLocking

Syntax:

C, C++: short **L862_ExecLocking** (short hPort, bool bLock);

V.Basic: function **L862_ ExecLocking** (ByVal hPort As Integer,
ByVal bLock As Boolean) As Integer

Purpose:

This function has to be called to command the card locking system.

The locking system can only be set to “locked” position when a card is seated.

Return value:

The positive code L862_OK is returned, when the lock system has been positioned at the desired state.

The value L862_NEGATIVE is returned, if the locking status is not conformed to the order.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

bLock

= TRUE: to get a card-locked position.

= FALSE to unlock the card

L862_ExecCapture

Syntax:

C, C++: short **L862_ExecCapture** (short hPort);

V.Basic: function **L862_ ExecCapture** (ByVal hPort As Integer,
ByVal bLock As Boolean) As Integer

Purpose:

This function has to be called to eject the card through the rear side of the reader

This function can only be executed when a card is under the reader’s control.

Return value:

The positive code L862_OK is returned, when the card has been successfully ejected.

The value L862_NEGATIVE is returned, if the card is present after the process timeout.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

L862_SetGreenLed / L862_SetRedLed

Syntax:

C, C++: short **L862_SetRedLed** (short hPort, short state);
short **L862_SetGreenLed** (short hPort, short state);

V.Basic: function **L862_SetRedLed** (ByVal hPort As Integer,
ByVal State As Integer) As Integer
function **L862_SetGreenLed** (ByVal hPort As Integer,
ByVal State As Integer) As Integer

Purpose:

This function has to be called to command the user led that is installed at the front of the reader (one bi-color led, or two different leds)

Return value:

The positive code L862_OK is returned, when the reader has acknowledged the command

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

State

= 0: to switch the led OFF

= 1: to switch the led ON

= 2: to begin flashing the led

L862_ReadDescription

Syntax:

C, C++: short **L862_ReadDescription** (short hPort, short DescCode, short *DataLen, char *pData);

V.Basic: function **L862_ReadDescription** (ByVal hPort As Integer, ByVal DescCode As Integer, ByRef DataLen As Integer, ByVal pdata As String) As Integer

Purpose:

This function permits to ask for some characteristics from the reader

Return value:

The positive code L862_OK is returned, when the reader returns the requested data.

A null or negative value is returned, if this function fails for any others reasons.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

DescCode:

Type of requested information:

- 0 (Desc_SERIAL): To get the reader’s serial number
- 1 (Desc_COPYRIGHT): To get the reader’s copyright information
- 2 (Desc_MODEL): To get the reader’s model description
- 3 (Desc_PCB): To get the reader’s electronic board version
- 4 (Desc_VERSION): To get the reader’s application version
- 5 (Desc_BOOT): To get the reader’s boot module version
- 6 (Desc_LOADER): To get the reader’s loader module version
- 7 (Desc_CONFIG): To read the reader’s configuration data block

Len:

Double pointer:

- At function call, this value has first to be set with the given “pData” buffer size.
- When function returns, this pointer will contains the returned DATA length.

This value shall not be null

pData:

Pointer to storage buffer to received the returned ASCII string, which will correspond to the requested description.

This pointer may not be NULL.

Note:

The function **L862_ReadVersion** can also be used to get the reader’s application version

L862_ReadStatus

Syntax:

C, C++: short **L862_ReadStatus** (short hPort, short *Status1)

short **L862_ReadStatusEx** (short hPort, short *Status1, short *Status2);

V.Basic: function **L862_ReadStatus** (ByVal hPort As Integer,
ByRef Status1 As Integer) As Integer

function **L862_ReadStatusEx** (ByVal hPort As Integer, ByRef Status1 As Integer,
ByRef Status2 As Integer) As Integer

Purpose:

This function has to be called to get one or two of the reader's status words.

The first status words inform about:

- the two switches status: Card position
- the reader's current modes: arming mode, data present
- the locking position
- the ICC status: activated or not
- the reader's configuration: Automatic modes, CTS/DTR mode
- the user-driven led status

The second status words inform about:

- the current ICC selection: User card or SAM
- Last ICC data transfer error status
-

see document "Command Reference Manual" for all status bits descriptions

Return value:

The positive code L862_OK is returned, when the reader has returned the status word

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Status

Returned status word

Note:

After executing this function, the following functions can be called to decode the status bits meaning:

- Get862_IsCardPresent
- Get862_IsCardSeated
- Get862_IsLocked
- Get862_IsArmed
- Get862_IsChipActive
- Get862_ChipSelection

L862_ReadCardPosition

Syntax:

C, C++: short **L862_ReadCardPosition** (short hPort, short *Position);

V.Basic: function **L862_ReadCardPosition** (ByVal hPort As Integer,
ByRef Position As Integer) As Integer

Purpose:

This function has to be called to get the reader's switches status, and so to get the Card position.

Return value:

The positive code L862_OK is returned, when the reader has returned the status word.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Position

Returned the card position

- 0: No card detected
- 1 : Card detected at front position, but not completely inserted
- 3: Card seated.

Note:

After executing this function, the following 2 functions

- Get862_IsCardPresent
- Get862_IsCardSeated

can also be used to get the card position.

L862_ReadIsoTrack

Syntax:

C, C++: short **L862_ReadIsoTrack** (short hPort, short Track, short Format,
short *Len, char * pData);

V.Basic: function **L862_ReadIsoTrack** (ByVal hPort As Integer,
ByVal Track as Integer, ByVal Format as Integer, ByRef Len As Integer,
ByVal pData As String) As Integer

Purpose:

This function has to be called to get the data decoded from the magnetic tracks during the last head pass. The calling of this function build data, supposing they are formatted conformed to ISO 7810 specifications:

- Track 1: 7 bits per char. (including 1 odd parity bit)
- Track 2 and 3: 5 bits per char. (including 1 odd parity bit)

Return value:

The positive code L862_OK is returned if correct data has been decoded

The null code L862_NEGATIVE is returned if returned data contains decoding errors, or no data have been decoded. In this case, only one char is returned as Data; and this character will be an usiRet_*ErrorCode.*

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

Track:

The number of the track to read: 1, 2 or 3.

Format

The format to be applied for conversion of decoded bits to ASCII data

- 0: standard format (depending on the track)
- 1: format ISO1: 7 bits per char
- 2: format ISO2: 5 bits per char
- 3: format ISO3: 5 bits per char

The parity bits will be checked during the conversion. Start / End characters must be found. The final LRC character must be correct too.

Len:

Double pointer:

- At function call, its value has to be set with the given “pData” buffer size.
- When function returns, this pointer will be set with the returned DATA length.

pData:

Pointer to storage buffer for the returned data. This magnetic data are converted into ASCII format.

L862_ReadCustomTrack

Syntax:

C, C++: short **L862_ReadCustomTrack** (short hPort, short Track, short Format, short *Len, char * pData);

V.Basic: function **L862_ReadCustoùTrack** (ByVal hPort As Integer, ByVal Track as Integer, ByVal Format as Integer, ByRef Len As Integer, ByVal pData As String) As Integer

Purpose:

This function has to be called to get the data decoded from the magnetic tracks during the last head pass. The calling of this function build data, which may be or not ISO.

No parity or LRC checking is perform. Because data are always filled with zeros, the first decoded 'one' becomes the beginning of the data.

Return value:

The positive code L862_OK is returned if correct data has been decoded

The null code L862_NEGATIVE is returned if no data have been decoded (data buffer is empty). In this case, only one char is returned as Data; and this character will be an usiRet_NO_DATA or usiRet_NO_MAG_CARD.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Track:

The number of the track to read: 1, 2 or 3.

Format

The format specifies the number of bits per character. This value must be included in the interval from 3 to 8. In this set of bits, the less significant bit (LSB) is coded first.

Len:

Double pointer:

- At function call, its value has to be set with the given "pData" buffer size.
- When function returns, this pointer will be set with the returned DATA length.

pData:

Pointer to storage buffer for the returned data. Magnetic data will be binary codes (not ASCII) and conformed to the given format. No conversion is performed.

Example: If 'Format' = 5

All returned data values V are: $00_h < V < 1F_h$

L862_ReadCorruptedTrack

Syntax:

C, C++: short **L862_ReadCorruptedTrack** (short hPort, short *Len, char * pData);

V.Basic: function **L862_ReadCorruptedTrack** (ByVal hPort As Integer,
ByRef Len As Integer, ByVal pData As String) As Integer

Purpose:

This function may be called just after getting an error from the function L862_ReadIsoTrack. In case ISO reading encounters decoding errors (bad parity, bad LRC, no Start or no End control-character found), the function **L862_ReadIsoTrack** returns the error code L862_NEGATIVE (Because the reader returns the error message usiRet_ERROR.

This function may be used to get the contents of this corrupted decoded data.

Return value:

The positive code L862_OK is returned if the data has been returned by the reader

A negative code is returned if this function fails

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

Len:

Double pointer:

- At function call, its value has to be set with the given “pData” buffer size.
- When function returns, this pointer will be set with the returned DATA length.

pData:

Pointer to storage buffer for the returned data. This magnetic data are converted into ASCII format.

L862_ChipSelect

Syntax:

C, C++: short **L862_ChipSelect** (short hPort, short iSam);

V.Basic: function **L862_ChipSelect** (ByVal hPort As Integer, ByVal iSam As Integer) As Integer

Purpose:

This function has to be called to select whose chip card or SAM will have to be selected (or addressed) for the future Chip communication functions.

When the reader starts (or resets) the Chip card is selected

Return value:

The positive code L862_OK is returned, when the reader acknowledges this order.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

iSam

Index of the ICC / SAM

- 0: the user inserted chip card (default selection)
- 1: the 1st SAM module
- 2: the 2nd SAM module
- *X*: the SAM module N. *X*

The reader’s application and configuration define the maximum value.

L862_ChipActivate

Syntax:

C, C++: short **L862_ChipActivate** (short hPort, short *Len, char * pATR);

short **L862_ChipActivateEx** (short hPort, short Spec, short *Len, char * pATR);

V.Basic: function **L862_ChipActivate** (ByVal hPort As Integer,
ByRef Len As Integer, ByVal pATR As String) As Integer

Purpose:

This function may be used to activate or reset the user chip card,one of the SAMs, or a memory card, conformed to ISO specifications.

Asynchronous Chip card (ICC / SAM), with microprocessor, are activated, with respect of payment system specification: EMV 2000.

Synchronous card could be activated with this command.

When a ICC (asynchronous card) is powered, it performs a cold reset. If it was already powered, then this function performs a warm reset.

When the asynchronous activation fails, the reader tries again, but with synchronous mode. If a memory card has been inserted, its ATR will be returned

The returned ATR will be 4-bytes length, when the card is a synchronous memory card (Siemens SLE, Gemplus…). The returned ATR will be 8-bytes length, if the card works with the protocol i²c (from Philips)

When a synchronous card is still powered, and this function is called again, then a error code returned. There's no "warm reset" with memory card.

Return value:

The positive code L862_OK is returned if the card has been activated, and a valid ATR frame has been returned by the card (via the reader)

The code L862_NEGATIVE is returned if no valid ATR has been returned by the card (or SAM). In this case, the chip is powered off.

A negative code is returned if the function call fails.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Len:

Double pointer:

- At function call, its value has to be set with the given "pATR" buffer size.
- When function returns, this pointer will be set with the returned ATR length.

Spec:

This option allows to precise the kind of specification: (see command ref manual for more precision)

- 0 : Only memory card will be activated (Synchronous mode only)
- 1 : Only µ-proc chip (ISO7816) will be activated (Asynchronous mode only)
- 2 : Only µ-proc chip that are EMV2000 compatible will be activated

pATR:

Pointer to storage buffer where the returned ATR data, given by the card, will be written.

The function **L862_ChipActivateEx** is only available on readers with firmware released after August 2004

L862_ChipDeactivate

Syntax:

C, C++: short **L862_ChipDeactivate** (short hPort);

V.Basic: function **L862_ChipDeactivate** (ByVal hPort As Integer) As Integer

Purpose:

This function may be used to deactivate and power off the selected chip card or SAM. (conformed to EMV 2000 specifications)

This function has also to be called to deactivate a memory card

Return value:

The positive code L862_OK is returned when the reader acknowledges this order

A negative code is returned if this function fails.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

L862_ChipRead

Syntax:

C, C++: short **L862_ChipRead** (short hPort, short cApduLen,
const char * cApdu, short rApduLen, char * rApdu);

V.Basic: function **L862_ChipRead** (ByVal hPort As Integer,
ByVal cApduLen As Integer, ByVal cApdu As String,
ByRef rApduLen As Integer, ByVal rApdu As String) As Integer

Purpose:

This function is recommended to be used, to communicate with asynchronous integrated chip card (ICC) or Secure application module (SAM); to exchange information with it.

An APDU command, be to send to the chip, is formatted as:

CLA	INS	P1	P2	Lc	…command DATA ….	Le

An APDU response, returned by the chip, is formatted as:

… response DATA …	SW1	SW2

These APDU has to be conformed to ISO7816-4 specification. The transport layer is performed automatically by the reader.

Return value:

The positive code L862_OK is returned if the response data has been correctly returned by the reader

The code L862_NEGATIVE is returned if no valid data has been returned. In this case, the chip will be automatically deactivated.

A negative code is returned if this function fails for another reason

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

cApduLen:

This value is the number of bytes of the "cApdu" message

cApdu

Pointer to the "cApdu" command buffer. The reader will have to send this command to the chip (or SAM) with respecting card protocol.

rApduLen:

Double pointer:

- At function call, its value has to be set with the given "rApdu" buffer size.
- When function returns, this pointer will be set with the returned DATA length.

rApdu:

Pointer to storage buffer where the returned "rApdu" data coming from the card will be written.

L862_ChipWrite

Syntax:

C, C++: short **L862_ChipWrite** (short hPort, short cApduLen,
const char * cApdu, short * SW);

V.Basic: function **L862_ChipWrite** (ByVal hPort As Integer,
ByVal cApduLen As Integer, ByVal cApdu As String,
ByRef SW As Integer) As Integer

Purpose:

This function is a second way to send commands or information to a card, when no data is expected as response.

This function has to be used, when only commands have to be sent to asynchronous integrated chip card (ICC) or Secure application module (SAM).

With this function, an APDU command, has to formatted as:

CLA	INS	P1	P2	Lc	…command DATA ….

The APDU response, expected from the chip, should not contain data, but only the status words:

SW1	SW2

Return value:

The positive code L862_OK is returned if only the reader has correctly returned the status word

The code L862_NEGATIVE is returned if no data has been returned. In this case, the chip will be automatically deactivated.

A negative code is returned if this function fails for another reason.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

cApduLen:

This value is the number of bytes of the “cApdu” message

cApdu

Pointer to the “cApdu” command buffer. The reader will have to send this command to the chip (or SAM) with respecting card protocol.

SW:

Pointer to storage buffer where the 2 status bytes will be written.

A correct execution is generally acknowledged by the card with SW = 9000_h

L862_MemSelectType

Syntax:

C, C++: short **L862_MemSelectType** (short hPort, short iType, short Size);

V.Basic: function **L862_MemSelectType** (ByVal hPort As Integer, ByVal iType As Integer, ByVal Size As Integer) As Integer

Purpose:

This function has to be called to select whose type of synchronous card will have to be used. So the appropriate card protocol is internally selected to correctly performed communication with the card.

When the reader starts (or resets) the Chip card is selected

Return value:

The positive code L862_OK is returned, when the reader acknowledges this order.

The code L862_NEGATIVE is returned if the given type code is unknown

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

iType

Index that identify the type of memory card:

- 0: Card with units: SLE4406, PCF2006, AT88SC06
- 1: Cards SLE4418, 4428, GMP8K
- 3: Cards SLE4432, 4442, PCF203x,
- 4: Cards using i²c protocol (<256 bytes)
- 5: Cards using i²c protocol (>256 bytes)

See the “Command reference manual” for more details about the list of the available card type

Size

This parameter is optional, it is only useful for some I²C card to specify the card page size.

See the “Command reference manual” to get more information

L862_MemTransfer

Syntax:

C, C++: short **L862_MemTransfer** (short hPort, short cApduLen,
const char * cApdu, short rApduLen, char * rApdu);

V.Basic: function **L862_MemTransfer** (ByVal hPort As Integer,
ByVal cApduLen As Integer, ByVal cApdu As String,
ByRef rApduLen As Integer, ByVal rApdu As String) As Integer

Purpose:

This function is recommended to be used, to communicate with synchronous card: Memory cards or I²C card, for reading or writing data

An APDU command, be to send to the reader, is formatted as:

CLA	INS	$Addr_{high}$	$Addr_{low}$	Lc	…command DATA ….	Le

An APDU response, returned by the reader, is formatted as:

… response DATA …	SW1	SW2

This format is used to keep compatibility with ISO7816-4 specifications and ICC card. The transport layer is performed automatically by the reader.

Return value:

The positive code L862_OK is returned if the response data has been correctly returned by the reader

The code L862_NEGATIVE is returned if the request could not be performed.

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

cApduLen:

This value is the number of bytes of the “cApdu” message

cApdu

Pointer to the “cApdu” command buffer. The reader will have to send this command to the chip (or SAM) with respecting card protocol.

rApduLen:

Double pointer:

- At function call, its value has to be set with the given “rApdu” buffer size.
- When function returns, this pointer will be set with the returned DATA length.

rApdu:

Pointer to storage buffer where the returned “rApdu” data coming from the card will be written.

3.5 Reading tool functions

After having perform the function L862_ReadStatus or L862_ReadStatusEx, the followings functions can be used to evaluate the returned status words.

These functions do **not** generate communication with the reader. They may only be used to get the meaning of some status bits.

Get862_IsCardPresent

Syntax:

C, C++: BOOL **Get862_IsCardPresent** (short hPort);

V.Basic: function **Get862_IsCardPresent** (ByVal hPort as Integer) as Boolean

Purpose:

Return the state of the 'Start switch'. This switch is placed at the front of the reader. If a card is partially or completely inserted, the switch is activated.

Return value

TRUE if a card is detected

FALSE when no card is detected

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Get862_IsCardSeated

Syntax:

C, C++: BOOL **Get862_IsCardSeated** (short hPort);

V.Basic: function **Get862_IsCardSeated** (ByVal hPort as Integer) as Boolean

Purpose:

Return the state of the 'End switch'. This switch is placed at the rear of the reader. The switch is activated when a card is completely inserted inside the reader. In this position, the locking system is authorized to be moved into the lock position; the chip can be activated.

Return value

TRUE if a card is seated inside the reader

FALSE when there's no card, or the card is not completely inserted

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Get862_IsLocked

Syntax:

C, C++: BOOL **Get862_IsLocked** (short hPort);

V.Basic: function **Get862_IsLocked** (ByVal hPort as Integer) as Boolean

Purpose:

Return the state of the 'locking switch'. This switch, inside the reader, is activated when the locking system is being at locked position to keep the inserted card inside the reader.

Return value

TRUE in the locked position

Else FALSE.

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Get862_IsArmed

Syntax:

C, C++: BOOL **Get862_IsArmed** (short hPort);

V.Basic: function **Get862_IsArmed** (ByVal hPort as Integer) as Boolean

Purpose:

Return the reader's functional mode. This mode is set when the reader receives the command usiCmd_ARM (using the function L862_ArmToRead). Then the reader is ready to decode data from magnetic stripes. This mode is automatically cleared after having read magnetic stripes, or by removing the card outside the reader, or by the use of the command usiCmd_ABORT.

Return value

TRUE if the "arm" mode is active

FALSE if the "arm" mode is inactive

Parameters:

hPort:

The handle of the communication device, given by "L862_Open".

Get862_ChipSelection

Syntax:

C, C++: short **Get862_ChipSelection** (short hPort);

V.Basic: function **Get862_IsChipSelection** (ByVal hPort as Integer) as Integer

Purpose:

Return the index of the current selected Chip card or SAM.

Return value

0 when the user chip card is selected (is the default selection)

1 when the SAM #1 is selected

2 when the SAM #2 is selected

…

N when the SAM #*N* is selected

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.

Get862_IsChipActive

Syntax:

C, C++: BOOL **Get862_IsChipActive** (short hPort);

V.Basic: function **Get862_IsChipActive** (ByVal hPort as Integer) as Boolean

Purpose:

Return the state of the current selected chip card or SAM.

Return value

TRUE if the current selected ICC (or SAM) is active (powered ON)

FALSE if the current selected ICC is powered OFF

Parameters:

hPort:

The handle of the communication device, given by “L862_Open”.