ASKCSC.DLL API Software Specifications

RELEASE HISTORY

Release Number	Date	Subject / Comments	Writer	approv.
4.08.13.25	2015/03/13	• Add DESFire_AuthenticateEV1 • DESFire_ChangeKey: complete update	GB	
4.08.12.24	2014/12/05	• DESFire_WriteRecord: parameters correction • Add CSC_SetSAMBaudratePPS. • Add EMVCo functions. • Add Calypso Rev3 functions. • Add Mifare ULC and Mifare UL EV1 functions. • Add DESFIRE_PrepareAuthentication • Host low level communication enhanced.	GB	
4.08.10.22	2012/09/12	• ST SR added in CSC_SearchCardExt. • Single shot search in CSC_SearchCardExt. • Description of “ASKCSC.ini” entries	GB	
4.08.9.19	2011/06/06	• Correction Class SRx Read / Write.	DF	
4.08.9.18	2011/05/04	• Add function Desfire • Add Class Mifare Plus SL3 • Add Class SRx • Update Class Mifare Classic. • Update Class system (ResetSam, Transparent, WriteEeprom ...) • Update GTML • Update CD97 • Update Generic	DF	
4.08.9.17	2010/03/11	• Add function Mifare SAM NXP	DF	
4.08.5	2008/03/04	• Correction on long frames Rx on USB and serial link	GB	PC
4.08.4	2007/01/09	• Restart PC/SC driver on Close (if PCSCDontStart=0 in ini file)	GB	FM
4.08.3	2006/10/03	• Log buffer extented to 512 bytes • PC/SC timeout set	GB	FM
4.08.2	2006/06/06	• MIFARE_xxx functions work with Mifare 4ko • Baudrate set with USB/serial converter using USB driver	GB	FM
4.08.1	2006/04/18	• Support of GEN4XX PCSC mode • iCSC_TransparentCommandConfig uses 01 20 & iCSC_TransparentCommand uses 01 21	GB	FM
4.07.4	2006/02/07	• USBLatencyTimer in ini file	GB	FM
4.07.3	2005/12/23	• Support of GEN4XX USB native mode in CSC_SearchCSC	GB	FM
4.07.2	2005/11/16	• Support of GEN4XX USB native mode	GB	FM
4.07	2005/06/13	• Add CSC_DesactiveCRC()	CC	FM
4.06	2005/01/04	• Add parallel port communication on Windows NT/2000/XP (as Windows 9x) • ResetCSC : don't check for 0x10 • COM name extended to \\.\COMx • Send data to CSC asynchronously • GTML_OpenSession (check frame length < 6 → < 5)	GB	FM
4.05	2003/09/10	• Add long frames management	BL	FM
4.05 β Version	2003/08/06	• Add mono search mode • Add CTx512x class • Add CSC_WriteSAMNumber command • Add CSC_TransparentCommand and CSC_TransparentCommandConfig	BL	FM
4.04	2003/01/22	• Final example modification	BL	

4.03	2002/01/02	• Correct CSC_SearchCardExt exporation • Change CTx512B status	BL	FM
4.02		• Add examples	BL	FM
4.02 β Version	2002/09/04	• Add CTx512B functions Add CSC_ISOCommandContact function Add CSC_SearchCardExt function Modify VerifyPIN function for clear mode Add PINStatus function Add MIFARE_Select function Restore correct parameters in some Mifare® and Certificate functions	BL	FM
4.01	2002/03/28	• Structure Definition	SM	FM
4.00	2002/02/18	• Add MIFARE® and special systems functions	SM	FM
3.11	2001/11/01	• Change order of parameters to the correct one in Csc_SearchCard()	SM	FM
3.10	2001/03/29	• Added CTx and certificat functions	CC	FM
2.0	2000/05/12	• Added GTML functions.	JLM	FM
1.2	1999/08/02	• Functions PIN correction	TJ	FM
1.1	1999/06/08	• Examples correction • Functions correction CD97_ReloadEP	TJ	FM
1.0	1999/05/19	• ASKCSC.DLL functions description • DLL use examples	TJ	FM

REFERENCE DOCUMENTS

Source	Document	Description
ASK GENXXX Dev Kit	RD-ST-08167-xx	ASK CSC - Coupler Software Interface_Gen5XX

Contents

Release History 2

1. Library Description 9

1.1. Objective 9

1.2. Files description 9

1.3. Library functions description 11

1.4. Document convention and Notation 12

FUNCTION NAME 12

1.5. Structures used in the DLL functions 13

sCARD_Status Structure 13

sCARD_Session Structure 13

sCARD_SecurParam Structure 13

sCARD_Search Structure 14

sCARD_SearchExt Structure 14

1.6 Constants used 15

Functions return 15

SAM/contact slots 15

search masks for CSC_SearchCardExt function 16

1.7. Library Specific Functions 17

CSC_Open 17

CSC_Close 17

CSC_WriteConfigEeprom 18

CSC_ReadConfigEeprom 19

CSC_GetUSBNumDevices 20

CSC_GetPCSCNumDevices 20

CSC_GetPCSCDeviceName 21

CSC_VersionDLL 22

CSC_AddCRC 23

CSC_SearchCSC 24

CSC_ChangeRS485Address 24

CSC_ChangeComSpeed 25

1.8. ASK coupler management functions 26

CSC_ResetCSC 26

CSC_VersionCSC 27

CSC_DesactiveCRC 28

CSC_CardStartSearch 29

CSC_CardStopSearch 29

CSC_CardFound 31

CSC_CardEnd 32

CSC_SearchCard 33

CSC_SearchCardExt 34

CSC_SendReceive 36

CSC_TransparentCommandConfig 37

CSC_TransparentCommandConfigExt 38

CSC_TransparentCommand 39

CSC_ISOCommand 41

CSC_ISOCommandContact 42

CSC_SelectSAM 43

CSC_ResetSAM 44

CSC_ResetSAMExt 44

CSC_ISOCommandSAM 46

CSC_ISOCommandSAMExt 47

CSC_SetSAMBaudratePPS 48

CSC_AntennaOFF 49

CSC_Switch_Led_Buz 50
CSC_SelectCID 51
CSC_SelectDIV 52
CSC_EHP_PARAMS 53
CSC_EHP_PARAMS_EXT 54
CSC_WriteSAMNumber 55

1.8. EMVCo functions 57
EMVCo_UserInterface 57
EMVCo_Contactless 58

1.8. Calypso Rev3 functions 60
CalypsoRev3_GetMode 60
CalypsoRev3_SetMode 60

1.9. GTML card management functions 61
GTML_SelectFile 61
GTML_Invalidate 62
GTML_Rehabilitate 63
GTML_ChangePIN 64
GTML_VerifyPIN 65
GTML_Increase 66
GTML_Decrease 67
GTML_ReadRecord 68
GTML_AppendRecord 69
GTML_UpdateRecord 70
GTML_WriteRecord 71
GTML_OpenSession 72
GTML_CloseSession 73
GTML_AbortSecuredSession 74

1.10. CD97 card management functions 75
CD97_SelectFile 75
CD97_StatusFile 76
CD97_Invalidate 76
CD97_Rehabilitate 77
CD97_ChangeKey 78
CD97_ChangeKeyExt 79
CD97_ChangePIN 80
CD97_ChangePINExt 81
CD97_VerifyPIN 82
CD97_VerifyPINExt 82
CD97_Increase 84
CD97_Decrease 85
CD97_ReadRecord 86
CD97_AppendRecord 87
CD97_UpdateRecord 88
CD97_WriteRecord 89
CD97_OpenSession 90
CD97_ OpenSessionExt 91
CD97_CloseSession 93
CD97_CloseSessionExt 94
CD97_AbortSecuredSession 95
CD97_SelectISOApplication 96
CD97_Purchase 97
CD97_GetEPStatus 98
CD97_ReloadEP 99
CD97_CancelPurchase 100

1.11. Variable class mapping 101
SelectFile 101
StatusFile 102
Invalidate 103
Rehabilitate 104
ChangePIN 105

VerifyPIN 106
PINStatus 107
Increase 108
Decrease 109
ReadRecord 110
ReadRecordMultiple 111
AppendRecord 112
ChangeKey 113
UpdateRecord 114
WriteRecord 115
OpenSession 116
OpenSessionExt 117
CloseSession 118
AbortSecuredSession 119
Lock_Unlock 120
Multi_Decrease 121
Multi_Increase 122
GetEPStatus_CD97 123
Purchase_CD97 124
ReloadEP_CD97 125
CancelPurchase_CD97 126
DecreaseLG 127
IncreaseLG 128
ReadBinary 129
UpdateBinary 129
WriteBinary 130
CalypsoRev3_SelectApplication 130
CheckCertificate 131
GiveCertificate 132

1.12. CTx card management functions 133
1.12.1. CTS256B functions 133
CTx_Active 133
CTx_Read 134
CTx_Update 135
CTx_Release 136
1.12.2. CTx512x functions 137
CTx512x_List (CTx512B only) 137
CTx512x_Select (CTx512B only) 139
CTx512x_Read 140
CTx512x_Update 141
CTx512x_Write 142
CTx512x_Halt 143
CTx512x_Authenticate (CTM512B only) 144
CTx512x_WriteKey (CTM512B only) 145

1.13. MIFARE® card functions 146
MIFARE_LoadReaderKeyIndex 146
MIFARE_ChangeKey 147
MIFARE_Select 148
MIFARE_Authenticate 149
MIFARE_Halt 150
MIFARE_ReadBlock 150
MIFARE_ReadSector 151
MIFARE_WriteBlock 153
MIFARE_DecrementValue 154
MIFARE_IncrementValue 155
MIFARE_BackUpRestoreValue 156
MIFARE_ReadMultipleBlock 157
MIFARE_SimpleWriteBlock 157
MIFARE_ReadSectorData 158
MIFARE_WriteSectorData 158

1.13. MIFARE® - SAM NXP card functions 160

MIFARE_SAMNXP_Authenticate 160
MIFARE_SAMNXP_Re-Authenticate 161
MIFARE_SAMNXP_ReadBlock 162
MIFARE_ SAMNXP_WriteBlock 162
MIFARE_ SAMNXP _ChangeKey 163
MIFARE_ SAMNXP _Increment 164
MIFARE_ SAMNXP _Decrement 164
MIFARE_ SAMNXP _BackUpValue 165
MIFARE_ SAMNXP _ResetAuthentication 165

1.13. MIFARE® PLUS card functions 166
MFP_SL3_Authentication 166
MFP_SL3_ResetAuthentication 167
MFP_SL3_ReadBlock 167
MFP_SL3_WriteBlock 168
MFP_SL3_ChangeKey 168
MFP_SL3_VirtualCardSupport 169
MFP_SL3_DeselectVirtualCard 169

1.13. SRx Family card functions 170
SRX_Active 170
SRX_ReadBlock 170
SRX_WriteBlock 171
SRX_Release 171
SRX_Read 172
SRX_Write 172

1.13. Desfire card functions 174
DESFIRE Status 174
DESFIRE_CreateApplication 175
DESFIRE_DeleteApplication 175
DESFIRE_SelectApplication 176
DESFIRE_FormatPICC 176
DESFIRE_GetApplicationIDs 176
DESFIRE_GetVersion 177
DESFIRE_GetFreeMem 178
DESFIRE_PrepareAuthentication 178
DESFIRE_Authenticate 178
DESFIRE_AuthenticateEV1 179
DESFIRE_CommitTransaction 180
DESFIRE_AbortTransaction 180
DESFIRE_ChangeKey 181
DESFIRE_ChangeKeySetting 182
DESFIRE_GetKeySetting 182
DESFIRE_GetKeyVersion 182
DESFIRE_ChangeFileSetting 183
DESFIRE_ClearRecordFile 183
DESFIRE_CreateBackUpDataFile 183
DESFIRE_CreateCyclicRecordFile 184
DESFIRE_CreateLinearRecordFile 184
DESFIRE_CreateStandardDataFile 185
DESFIRE_CreateValueFile 185
DESFIRE_Credit 186
DESFIRE_Debit 187
DESFIRE_DeleteFile 187
DESFIRE_GetFileID 187
DESFIRE_GetFileSetting 188
DESFIRE_GetValue 188
DESFIRE_LimitedCredit 189
DESFIRE_ReadData 189
DESFIRE_WriteData 190
DESFIRE_ReadRecord 190
DESFIRE_WriteRecord 191
DESFIRE_SamGetVersion 191

DESFIRE_SamSelectApplication 192
DESFIRE_SamLoadInitVector 192
DESFIRE_SamGetKeyEntry 192
DESFIRE_SamGetKucEntry 193
DESFIRE_SamDisableCrypto 193

1.13. Mifare Ultralight C and Mifare Ultralight EV1 functions 194
MFUL_Identify 194
MFUL_Read 195
MFUL_Write 196
MFULC_Authenticate 197
MFULC_WriteKeyFromSAM 198
MFULEV1_PasswordAuthenticate 199
MFULEV1_CreateDiversifiedPasswordandPACK 200
MFULEV1_ReadCounter 201
MFULEV1_IncrementCounter 201
MFULEV1_GetVersion 202
MFULEV1_CheckTearingEvent 203

2. DLL Functions use example 204

1. LIBRARY DESCRIPTION

1.1. Objective

The library's objective is to provide user with tools to create any application using the ASK coupler and to facilitate tests of the different CD97 and GTML cards functions. This library is provided with sources.

1.2. Files description

ASKCSC.DLL Version 4.xx.xx.xx	
Files Name	Description
ASKCSC.DLL	DLL
ASKCSC.INI	Configuration file
ASKCSC.LIB	Import Library
ASKCSC.H	Library prototypes functions header file
CSC_DEF.H	Constants definition File
Sources Files	Description
CSC_ORD.C	Order Coupler Procedures Sources (developed in ANSI C)
CSC_ORD.H	CSC_ORD.C functions prototypes file
WINCSC.C	Windows COM port functions sources
WINCSC.H	WINCSC.C functions prototypes file
ASKCSC.C	Library exported functions sources
ASKCSC.DEF	Library exported functions prototype
Example file	Description
TEST.C	Coupler test program (WIN32 console mode)

The “ASKCSC.INI” file is used to configure the DLL behavior. See below a file example:

; ASKCSC.INI

; This file must be located in the same directory as the module using ASKCSC.DLL.

[Configuration]

; used to reduce output flow. Can be necessary, for coupler used in terminals, trough transparent application on some terminals.

SlowFrame=0

; use soft reset for GEN5XX USB CDC (avoid hard reset, loosing virtual com port)

SoftReset=0

; allow the FTDI latency timer (only on GEN4XX USB) to be changed from 16 milliseconds to any value from 1 to 255 milliseconds

USBLatencyTimer=2

; do not restart PC/SC driver on CSC_Close

PCSCDontRestart=0

; preserve CPU usage on host communication. Suitable for most operation. unrecommended on some test suites

PreserveCPUUsage=0

; no retry on timeout host

NoRetryOnHostTimeout=0

1.3. Library functions description

This library is made up of 3 different groups of functions

- The library specific functions (Ex. CSC_Open, CSC_Close, CSC_VersionDLL)
- The coupler management functions (Ex. CSC_ResetCSC, CSC_SendReceive)
- The GTML card management functions (Ex. GTML_ReadRecord, GTML_Increase)
- The CD97 card management functions (Ex. CD97_ReadRecord, CD97_Increase, CD97_ChangeKey)
- The variable class mapping functions (Ex. OpenSession, ReadRecord)
- The CTx cards management functions (Ex. CTx512x_List, CTx_Read, CTx512x_Update)
- The Mifare® cards management functions (Ex. MIFARE_Authenticate, MIFARE_ReadSector)

1.4. Document convention and Notation

This manual uses the following typographic conventions

FUNCTION NAME

Description: Function execution actions

Syntax: C language function prototype

Parameters:

io	type	**name**	description

io:

- « `I` » = parameters IN
- « `O` » = parameters OUT

type:

Parameter Type

name:

Parameter Name

description:

Parameter description

Return: Function Return value

Value	designation

See also: Other functions name

Example : Use example.

1.5. Structures used in the DLL functions

List of the Structures

sCARD_Status Structure

Description: status for card functions

Parameters: List of members

BYTE	Code	status Code
BYTE	Byte1	status word 1
BYTE	Byte2	status word 2

sCARD_Session Structure

Description:For the parameters returned after an open session

Parameters: List of members

BYTE	NbApp	Number of Application
SHORT	Path[128]	Path of the Applications
BYTE	Data[29]	Data record

sCARD_SecurParam Structure

Description:For generic class set (parameters necessary for security process)

Parameters: List of members

BYTE	AccMode	Acces Mode
BYTE	SID	Short ID
WORD	LID	Long ID
BYTE	NKEY	Number of Key (SAM)
BYTE	RFU	Reserved for the KVC

sCARD_Search Structure

Description: For list of protocol search (Validation of the protocol required)

Parameters: List of members

BYTE	CONT	Contact Protocol Mode
BYTE	ISOB	ISO B Protocol Mode
BYTE	ISOA	ISO A Protocol Mode
BYTE	TICK	Ticket Protocol Mode
BYTE	INNO	Innovatron Protocol Mode

sCARD_SearchExt Structure

Description: For list of protocol search (Validation of the protocol mask required). The value corresponds to the search ratio.

Parameters: List of members and search ratio range.

BYTE	CONT	Contact Protocol Mode	0x00 to 0x03
BYTE	ISOB	ISO B Protocol Mode	0x00 to 0x03
BYTE	ISOA	ISO A Protocol Mode	0x00 to 0x03
BYTE	MIFARE	MIFARE Protocol Mode	0x00 to 0x03
BYTE	TICK	Ticket Protocol Mode	0x00 to 0x03
BYTE	INNO	Innovatron Protocol Mode	0x00 to 0x03
BYTE	MV4k	MV4000 Protocol Mode	0x00 to 0x03
BYTE	MV5k	MV5000 Protocol Mode	0x00 to 0x03
BYTE	MONO	For a single-shot search	0x00 or 0x01
BYTE	SRX	ST SR Protocol Mode	0x00 to 0x03

Example:

for searching ISOA, ISOB and MV4k cards with the respective ratios 1,2 and3 :

```
sCARD_SearchExt        Search;
Search.CONT=0x00;
Search.ISOB=0x02;
Search.ISOA=0x01;
Search.TICK=0x00;
Search.INNO=0x00;
Search.MIFARE=0x00;
Search.MV4k=0x03;
Search.MV5k=0x00;
```

1.6 Constants used

Functions return

Description: values returned by the DLL functions

Parameters: List of values

DWORD	RCSC_Ok	0x8001
DWORD	RCSC_OpenCOMError	0x8002
DWORD	RCSC_NoAnswer	0x8003
DWORD	RCSC_CheckSum	0x8004
DWORD	RCSC_Fail	0x8005
DWORD	RCSC_CardNotFound	0x8006
DWORD	RCSC_AntennaFails	0x8007
DWORD	RCSC_Timeout	0x8008
DWORD	RCSC_DataWrong	0x8009
DWORD	RCSC_Overflow	0x800A
DWORD	RCSC_ErrorSAM	0x800B
DWORD	RCSC_CSCNotFound	0x800C
DWORD	RCSC_BadATR	0x800D
DWORD	RCSC_TXError	0x800E
DWORD	RCSC_WarningVersion	0x800F
DWORD	RCSC_SelectSAMError	0x8010
DWORD	RCSC_UnknownClassCommand	0x8011
DWORD	RCSC_InputDataWrong	0x8012

SAM/contact slots

Description: SAM slots and contact slot constants. (The contact slot is slot number 4)

Parameters: List of values

BYTE	SAM_CURRENT	0x00
BYTE	SAM_SLOT_1	0x01
BYTE	SAM_SLOT_2	0x02
BYTE	SAM_SLOT_3	0x03
BYTE	SAM_SLOT_4	0x04
BYTE	CONTACT_SLOT	0x04

search masks for CSC_SearchCardExt function

Description: search masks for CSC_SearchCardExt function : *search_mask*, the 2nd argument, is constituted of the bit-to-bit OR between the masks corresponding to the type of cards searched.

Parameters: List of values

DWORD	SEARCH_MASK_CONT	0x0001
DWORD	SEARCH_MASK_ISOB	0x0002
DWORD	SEARCH_MASK_ISOA	0x0004
DWORD	SEARCH_MASK_TICK	0x0008
DWORD	SEARCH_MASK_INNO	0x0010
DWORD	SEARCH_MASK_MIFARE	0x0020
DWORD	SEARCH_MASK_MV4K	0x0040
DWORD	SEARCH_MASK_MV5K	0x0080
DWORD	SEARCH_MASK_MONO	0x0100
DWORD	SEARCH_MASK_SRX	0x0200

Example:

for searching ISOA, ISOB and MV4k cards :
DWORD search_mask;
search_mask=SEARCH_MASK_ISOB | SEARCH_MASK_ISOA | SEARCH_MASK_MV4K;

1.7. Library Specific Functions

CSC_Open

Description: This function opens the PC communication port, this procedure must be called before the other functions.

Syntax: DWORD WINAPI ***CSC_Open*** (LPSTR **ComName**) ;

Parameters:

I	LPSTR	**ComName**	PC communication port name (Ex: 'COMx' or 'LPTx')

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails

See also: CSC_Close

Example:

```
DWORD        ret;
ret=CSC_Open("COM2");
```

CSC_Close

Description: This function close the PC communication port.

Syntax: void WINAPI ***CSC_Close*** (void) ;

Parameters: None

Return: None

See also: CSC_Open

example :

```
CSC_Close();
```

CSC_WriteConfigEeprom

Description: Writes in the EEPROM configuration.

Syntax: DWORD WINAPI **CSC_WriteConfigEeprom**(BYTE **Index**,
BYTE **Value**,
BYTE ***Status**);

Parameters:

I	BYTE	**Index**	Index (1 byte) $01 : Value = serial (RS232/TTL/RS485) baud rate divider =1382400 / BAUDRATE $02 : Not relevant for GEN5xx : kept for compatibility with other products $03 : Not relevant for GEN5xx : kept for compatibility with other products $04 : Value = default SAM Number $05 : Field off CTx : turn on the field before CTx command and turn off the field during (Value * 1 ms) after CTx command. 0x00 and 0xFF disable field management on CTx $06 : Auto Led management enabled if Value = 1. Leds are managed by firmware (red = power on, orange = field on, green = reader/card communication. $07 : Not significant on GEN5xx $08 : Host communication frame padding : module 62 byte padding if Value = 62. $09 : ISO14443-4 number of retries. $0A : Delay between retries (ms). $0B : default RX RF speed at reset (00=106, 01=212, 02=424, 03=847 kb/s). $0C : default RX RF speed at reset (00=106, 01=212, 02=424, 03=847 kb/s). $0D : SAM reset at coupler reset (0=no reset) $0E : AUX Pin signal $0F : High baud rate ISO14443-A gain (00=20, 01=24, 02=31, 03=35 dB) $10 : Last Slot switch test (1=yes (CAM), other = no (SAM)) $11 : Strict ISO14443-3B timeout (1=strict check, other = no strict check, same as GEN3XX) $12 : Strict ISO14443-4B timeout (1=strict check, other = no strict check, same as GEN3XX) $13 : Delay after REQ/Select (0 or FF : no delay, same as GEN3XX, other = delay in ms) $14 : Unconditional Mifare selection before authentication (if value=1) $15 : Not significant on GEN5XX $16 : Custom Frame Waiting Time (Value * 10 ms, 00 or FF = no custom FWT) $17 : ISO14443-4 retries on PICC timeout (if value=1)
I	BYTE	**Value**	Value (1 byte)
O	BYTE	***Status**	Status of the operation (1 byte) $00 : Failure $01 : Success

Return: Return value

RCSC_Ok	The function succeeds
RCSC_Fail	The function fails

CSC_ReadConfigEeprom

Description: Read the value at the Index EEPROM.

Syntax: DWORD WINAPI **CSC_ReadConfigEeprom**(BYTE **Index**,
BYTE ***Status**,
BYTE ***Value**);

Parameters:

I	BYTE	**Index**	Index (1 byte) $01 : Value = serial (RS232/TTL/RS485) baud rate divider =1382400 / BAUDRATE $02 : Not relevant for GEN5xx : kept for compatibility with other products $03 : Not relevant for GEN5xx : kept for compatibility with other products $04 : Value = default SAM Number $05 : Field off CTx : turn on the field before CTx command and turn off the field during (Value * 1 ms) after CTx command. 0x00 and 0xFF disable field management on CTx $06 : Auto Led management enabled if Value = 1. Leds are managed by firmware (red = power on, orange = field on, green = reader/card communication. $07 : Not significant on GEN5xx $08 : Host communication frame padding : module 62 byte padding if Value = 62. $09 : ISO14443-4 number of retries. $0A : Delay between retries (ms). $0B : default RX RF speed at reset (00=106, 01=212, 02=424, 03=847 kb/s). $0C : default RX RF speed at reset (00=106, 01=212, 02=424, 03=847 kb/s). $0D : SAM reset at coupler reset (0=no reset) $0E : AUX Pin signal $0F : High baud rate ISO14443-A gain (00=20, 01=24, 02=31, 03=35 dB) $10 : Last Slot switch test (1=yes (CAM), other = no (SAM)) $11 : Strict ISO14443-3B timeout (1=strict check, other = no strict check, same as GEN3XX) $12 : Strict ISO14443-4B timeout (1=strict check, other = no strict check, same as GEN3XX) $13 : Delay after REQ/Select (0 or FF : no delay, same as GEN3XX, other = delay in ms) $14 : Unconditional Mifare selection before authentication (if value=1) $15 : Not significant on GEN5XX $16 : Custom Frame Waiting Time (Value * 10 ms, 00 or FF = no custom FWT) $17 : ISO14443-4 retries on PICC timeout (if value=1)
O	BYTE	***Status**	Status of the operation (1 byte) $00 : Failure $01 : Success
O	BYTE	***Value**	Value (1 byte)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_Fail	The function fails

CSC_GetUSBNumDevices

Description: This function returns the number of CSC USB devices

Syntax: DWORD **CSC_GetUSBNumDevices** (DWORD ***NumDevices**) ;

Parameters:

I	DWORD	***NumDevices**	number of CSC USB devices

Return: Return value

RCSC_Ok	The function succeeds
RCSC_Fail	The function fails

Note : this function returns the number of CSC USB devices that are managed by the USB native driver FTD2XX.SYS. It will not count the device that are managed by a virtual communication port driver as an emulated COMx port.

CSC_GetPCSCNumDevices

Description: This function returns the number of CSC PCSC devices

Syntax: DWORD **CSC_GetPCSCNumDevices** (DWORD ***NumDevices**) ;

Parameters:

I	DWORD	***NumDevices**	number of CSC PCSC devices

Return: Return value

RCSC_Ok	The function succeeds
RCSC_Fail	The function fails

Note : this function returns the number of CSC PCSC devices that are managed by the ASK PCSC driver ASKPCSC.SYS.

CSC_GetPCSCDeviceName

Description: This function returns the name of a CSC PCSC device

Syntax: DWORD **CSC_GetPCSCDeviceName** (DWORD **DeviceNumber,** char * **sName**) ;

Parameters:

I	DWORD	**DeviceNumber**	CSC PCSC device number (0 to n)
O	char	***sName**	Buffer for CSC PCSC device name

Return: Return value

RCSC_Ok	The function succeeds
RCSC_Fail	The function fails

CSC_VersionDLL

Description: This Function retrieves the DLL version

Syntax: DWORD WINAPI ***CSC_VersionDLL*** (LPSTR **Version**) ;

Parameters:

O	LPSTR	**Version**	Pointer to the buffer to receive the null-terminated string containing the DLL Version

Return: DLL Version

0xVVRR (16 bits)	VV = version (If 01 = version 1) RR = release (If 05 = release 05) so Version 1.05

See Also: CSC_VersionCSC

Example:

```
DWORD       ret;
BYTE        version[VERSION_LN];
ret=CSC_VersionDLL(version);
```

CSC_AddCRC

Description: Processes and adds the CRC at the end of a command.

Syntax: DWORD WINAPI ***CSC_AddCRC*** (BYTE* **Buf**, LPDWORD **Len**) ;

Parameters:

I	BYTE*	**Buf**	Frame without CRC
I	LPDWORD	**Len**	Length of the frame without CRC
O	BYTE*	**Buf**	Frame with CRC retrieve by the function
O	LPDWORD	**Len**	New Length of the frame with CRC

Return: Return value

RCSC_Ok	The function succeeds
RCSC_Fail	The function fails
RCSC_Overflow	Buffer overflow

See also: CSC_SendReceive

Example:

```
DWORD   ret;
DWORD   LnCRC=0x0A;  // length without CRC, then with CRC after CSC_AddCRC
BYTE    BufIn[0x0C] = {0x80,0x07,0x01,0x03,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00};
            // frame without CRC calculated, but included (CRC at '0x00,0x00')

ret=CSC_AddCRC(BufIn,&LnCRC);
// now, LnCRC=0x0C. The previous '0x00,0x00' at the end of BufIn have been replaced by the good CRC value
```

CSC_SearchCSC

Description: Searches for an ASK coupler on any PC serial port. If a coupler is found the CSC_Open function is executed and the RESET coupler is executed.

Syntax: DWORD WINAPI ***CSC_Search*** (void) ;

Parameters: None

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_CSCNotFound	Coupler not found

See also: CSC_Open, CSC_Close

Example :

```
DWORD        ret;
ret=CSC_SearchCSC();
```

CSC_ChangeRS485Address

Description : Change command format to RS485 mode taking into account a RS485 Bus address from 1 to 15.

The CSC RS485 mode change is performed is the address passed in parameter is comprise between 1 to 15, the change to value 0 get back to the CSC RS232 format.

Syntax : DWORD WINAPI ***CSC_ChangeRS485Address*** (BYTE **RS485Addr**) ;

Parameters :

I	BYTE	**RS485Addr**	RS485 Bus Address from 1 to 15

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_CSCNotFound	Coupler not found

CSC_ChangeComSpeed

Description : Change the DLL communication speed. Default value is 115200 Bauds.

Syntax : DWORD WINAPI ***CSC_ChangeComSpeed*** (WORD **NewSpeed**) ;

Parameters :

I	WORD	**NewSpeed**	Serial Port communication speed 9600 to 115200
O	LPBYTE	**Status**	$0 = done, other = Error in input parameter

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_CSCNotFound	Coupler not found

1.8. ASK coupler management functions

CSC_ResetCSC

Description: Initializes the ASK coupler (calc CRC by default)

Syntax: DWORD WINAPI ***CSC_ResetCSC*** (void) ;

Parameters: None

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error

See also: CSC_VersionCSC

Example :

```
DWORD        ret;
ret=CSC_ResetCSC();
```

CSC_VersionCSC

Description: Function retrieves the ASK coupler version

Syntax: DWORD WINAPI ***CSC_VersionCSC*** (LPSTR **Version**) ;

Parameters:

o	LPSTR	**Version**	Pointer to the buffer to receive the null-terminated string containing the ASK coupler Version

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_TXError	transmission error
RCSC_WarningVersion	Caution ! This coupler does not support the GTML class.

See also: CSC_VersionDLL

Example:

```
BYTE        version[VERSION_LENGTH];
DWORD       ret;
ret = CSC_VersionCSC(version);
```

CSC_DesactiveCRC

Description: Function actives (or desactives) the CRC and retrieves the ASK coupler version

Syntax: DWORD WINAPI ***CSC_DesactiveCRC*** (BYTE Type, LPSTR **Version**) ;

Parameters:

I	BYTE	**Type**	Type of frame: 0xFF (desactive the CRC), 0x00 (active the CRC)
O	LPSTR	**Version**	Pointer to the buffer to receive the null-terminated string containing the ASK coupler Version

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_TXError	transmission error
RCSC_WarningVersion	Caution ! This coupler does not support the GTML class.

See also: CSC_VersionDLL

Example:

```
BYTE        version[VERSION_LENGTH];
DWORD       ret;
ret = CSC_DesactiveCRC(255, version);
```

CSC_CardStartSearch

Description: Start card search

Syntax: DWORD WINAPI ***CSC_CardStartSearch*** (void) ;

Parameters: None

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error

See also: CSC_CardStopSearch, CSC_CardFound, CSC_CardEnd

Example:

```
DWORD        ret;
ret = CSC_CardStartSearch();
```

CSC_CardStopSearch

Description: Stop card search

Syntax: DWORD WINAPI ***CSC_CardStopSearch*** (void) ;

Parameters: None

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error

See also: CSC_CardStartSearch, CSC_CardFound, CSC_CardEnd

Example:

```
DWORD        ret;
```

```
ret = CSC_CardStopSearch();
```

CSC_CardFound

Description: Indicates if the ASK coupler found a card.

Syntax: DWORD WINAPI ***CSC_CardFound*** (BYTE* **lpATR**,
LPDWORD **lpcbATR**) ;

Parameters:

o	BYTE*	**lpATR**	ATR value
o	LPDWORD	**lpcbATR**	Length of the ATR value

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_CardNotFound	The coupler did not find card
RCSC_TXError	transmission error

Example:

```
DWORD       ret;
BYTE        atr[ATR_LENGTH_MAX];
DWORD       atrLength;
ret = CSC_CardFound(atr,&atrLength);
```

CSC_CardEnd

Description: Stops the communication with the card

Syntax: DWORD WINAPI ***CSC_CardEnd*** (void) ;

Parameters: None

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_CardStopSearch, CSC_CardStartSearch, CSC_AntennaOFF

Example :

```
DWORD        ret;
ret=CSC_CardEnd();
```

CSC_SearchCard

Description : Search a card or a ticket in different mode.

Syntaxe : DWORD WINAPI ***CSC_SearchCard*** (sCARD_Search **Search**,
BYTE **Forget,**
BYTE **TimeOut,**
LPBYTE **COM,**
LPDWORD **lpcbATR**,
BYTE* **lpATR**) **;**

Paramètres :

I	SCARD _Search	**Search**	Configuration structure of card research mode.
I	BYTE	**Forget**	Forget the serial number of the last card : ✓ Don't forget the last serial number : 0x00 ✓ Forget the last serial number : 0x01
I	BYTE	**TimeOut**	Card research delay = TimeOut x 10 ms
O	LPBYTE	**COM**	Communication mode found: ✓ Innovatron mode : 0x03 ✓ ISO 14443-B mode : 0x04 ✓ ISO 14443-A or MIFARE® mode : 0x05 or 0x08 ✓ Ticket CTS or CTM mode : 0x06 ✓ Contact mode : 0x07 ✓ TimeOut : 0x6F
O	LPDWORD	**lpcbATR**	Length of card response(see CSC interface for details).
O	BYTE*	**lpATR**	Response buffer value (see CSC interface for details).

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_CardStopSearch, CSC_AntennaOFF

Note : For further compatibility, the Search parameter values should be lower than 0x03.

Example:

```
DWORD           ret;
sCARD_Search    Search;
BYTE            COM;
BYTE            atr[ATR_LENGTH_MAX];
DWORD           atrLength;

Search.CONT=0x00;
Search.ISOB=0x02;
Search.ISOA=0x01;
Search.TICK=0x00;
Search.INNO=0x00;
ret=CSC_SearchCard(Search,0x01,0x44,&COM,&atrLength,atr);
```

CSC_SearchCardExt

Description : Search a card or a ticket in different mode – extension of CSC_SearchCard for MV4k, MV5k and Mifare® cards.

Syntaxe :

DWORD WINAPI ***CSC_SearchCardExt*** (sCARD_SearchExt * ***search***,
DWORD ***search_mask,***
BYTE **Forget,**
BYTE **TimeOut,**
LPBYTE **COM,**
LPDWORD **lpcbATR**,
BYTE* **lpATR**) **;**

Paramètres :

I	SCARD _SearchExt *	**search**	Pointer to the configuration structure of card search mode.
I	DWORD	***search_mask***	mask specifying the types of cards searched, constituted of the bit-to-bit OR between the masks corresponding to the type of cards searched
I	BYTE	**Forget**	Forget the serial number of the last card : ✓ Don't forget the last serial number : 0x00 ✓ Forget the last serial number : 0x01
I	BYTE	**TimeOut**	Card search delay = TimeOut x 10 ms
O	LPBYTE	**COM**	Communication mode found: ✓ Innovatron mode : 0x03 ✓ ISO 14443-B mode (ASK card) : 0x04 ✓ MIFARE® mode : 0x05 ✓ Ticket CTS or CTM mode : 0x06 ✓ Contact mode : 0x07 ✓ ISO 14443-A mode : 0x08 ✓ ISO 14443-B mode (non-ASK card) : 0x09 ✓ MV4k mode : 0x0A ✓ MV5k mode : 0x0B ✓ TimeOut : 0x6F
O	LPDWORD	**lpcbATR**	Length of card response(see CSC interface for details).
O	BYTE*	**lpATR**	Response buffer value (see CSC interface for details).

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_CardStopSearch, CSC_AntennaOFF

Example: (Search ISOB, ISOA and MV4k cards)

```
DWORD           ret;
sCARD_SearchExt SearchExt;
DWORD           search_mask;
BYTE            COM;
BYTE            atr[ATR_LENGTH_MAX];
DWORD           atrLength;

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x01;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x00;
SearchExt.MV4k=0x03;
SearchExt.MV5k=0x00;
search_mask=SEARCH_MASK_ISOB | SEARCH_MASK_ISOA | SEARCH_MASK_MV4K;

ret=CSC_SearchCardExt(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
```

CSC_SendReceive

Description: Sends a frame command directly to the coupler and retrieves the answer if this one happens before 'Timeout' milliseconds.

Syntax: DWORD WINAPI ***CSC_SendReceive*** (DWORD **Timeout**,
BYTE* **BufIN**,
DWORD **LnIN**,
BYTE* **BufOUT**,
LPDWORD **LnOUT**) ;

Parameters:

I	DWORD	**Timeout**	Timeout delay in milliseconds
I	BYTE*	**BufIN**	Frame to send
I	DWORD	**LnIN**	Length of the frame to send
O	BYTE*	**BufOUT**	Received Frame
O	LPDWORD	**LnOUT**	Length of the received frame

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_CardStartSearch

Example:

```
DWORD       ret;
DWORD       LnIn=0x0C;      // frame length, including CRC
DWORD       LnCRC;          // length without CRC, then with CRC after CSC_AddCRC
DWORD       LnOut;
BYTE        BufIn[0x0C] = {0x80,0x07,0x01,0x03,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00};
                    // frame without CRC calculated, but included (CRC at '0x00,0x00')
BYTE        BufOut[LN_OUT_MAX];

LnCRC=LnIn-2;
ret=CSC_AddCRC(BufIn,&LnCRC);
ret=CSC_SendReceive(2000, BufIn,LnIn,BufOut, &LnOut);
```

CSC_TransparentCommandConfig

Description: Configures the settings of "CSC_TransparentCommand".

Syntax: DWORD ***CSC_TransparentCommandConfig***(BYTE ***ISO***,
BYTE ***addCRC***,
BYTE ***checkCRC***,
BYTE ***field***,
BYTE* ***configISO***,
BYTE* ***configAddCRC***,
BYTE* ***configCheckCRC***,
BYTE* ***configField***);

Parameters:

I	BYTE	***ISO***	Mode setting : $00 : return the current configuration $01 : select ISOB $02 : select ISOA
I	BYTE	***addCRC***	$01 : the CRC will be computed and added to the frame else : nothing to add, the frame is sent directly
I	BYTE	***checkCRC***	$01 : the CRC of the frame received needs to be checked else : nothing to check
I	BYTE	***field***	$01 : the field will be switched ON when sending the frame else : no modification of the field
O	BYTE*	***configISO***	Current mode delected : $01 : ISOB selected $02 : ISOA selected $FF : wrong protocol asked
O	BYTE*	***configAddCRC***	current configuration (same values as input)
O	BYTE*	***configCheckCRC***	current configuration (same values as input)
O	BYTE*	***configField***	current configuration (same values as input)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

CSC_TransparentCommandConfigExt

Description: Configures the settings of "CSC_TransparentCommand".

Syntax: DWORD ***CSC_TransparentCommandConfigExt***(BYTE ***ISO***,
BYTE **addCRC**,
BYTE **checkCRC**,
BYTE **addParity**,
BYTE **checkParity**,
BYTE **numBitLastByte**,
BYTE **byPassISOA**,
BYTE **field**,
WORD **timeOut**,
BYTE* **configISO**,
BYTE* **configAddCRC**,
BYTE* **configCheckCRC**,
BYTE* **configAddParity**,
BYTE* **configCheckParity**,
BYTE* **configNumBitLastByte**,
BYTE* **configByPassISOA**,
BYTE* **configField,**
WORD* **configTimeOut**);

Parameters:

I	BYTE	***ISO***	0x00 : for getting the current config 0x01 : for selecting ISOB 0x02 : for selecting ISOA 0x03 : for selecting Felica (only Gen5xx)
I	BYTE	***addCRC***	$01 : the CRC will be computed and added to the frame else : nothing to add, the frame is sent directly
I	BYTE	***checkCRC***	$01 : the CRC of the frame received needs to be checked else : nothing to check
I	BYTE	***addParity***	$01 : the Parity will be computed and added to the frame else : nothing to add, the frame is sent directly
I	BYTE	***checkParity***	$01 : the Parity of the frame received needs to be checked else : nothing to check
I	BYTE	***numBitLastByte***	Number of bits of the last byte that shall transmitted 0 to 7 (1 byte)

I	WORD	***timeOut***	TimeOut Allowed for answer 0 to 2000 ms (default 456 ms) (2 bytes)
I	BYTE	***field***	$01 : the field will be switched ON when sending the frame else : no modification of the field
O	BYTE	****configISO***	0x01 : ISOB selected 0x02 : ISOA selected 0x03 : Felica selected 0xFF : wrong protocol asked
O	BYTE	****configAddCRC***	current configuration (same values as input)
O	BYTE	****configCheckCRC***	current configuration (same values as input)
O	BYTE	****configAddParity***	current configuration (same values as input)
O	BYTE	****configCheckParity***	current configuration (same values as input)
O	BYTE	****configNumBitLastByte***	current configuration (same values as input)
O	BYTE	****configByPassISOA***	current configuration (same values as input)
O	BYTE	****configField***	current configuration (same values as input)
O	WORD	**** configTimeOut***	current configuration (same values as input)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

CSC_TransparentCommand

Description: Sends and receives a transparent command, as previously configured with the CSC_TransparentCommandsConfig function.

Syntax: DWORD ***CSC_TransparentCommand*** (BYTE* ***bufIn***,
DWORD ***lnIn***,
BYTE* ***status***,
DWORD* ***lnOut***,
BYTE* ***bufOut***)

Parameters:

I	BYTE*	***bufIn***	Data to send
I	DWORD	***lnIn***	Length of the data to send
O	BYTE*	***status***	Status returned : $01 : CRC checked successfully (if asked) $FF : wrong CRC (if asked to be checked) $00 : CRC not checked
O	DWORD*	***lnOut***	Length of the data received
O	BYTE*	***bufOut***	Data received

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        configISO;
BYTE        configAdd;
BYTE        configCheck;
BYTE        configField;
DWORD       lnIn = 2;
BYTE        dataToSend[2] = {0x30,0x0A};
BYTE        status;
DWORD       lnOut;
BYTE        bufOut[MAX_ANSWER_LENGTH];

ret = CSC_TransparentCommandConfig(0x02, 0x01, 0x01, &configISO, &configAdd, &configCheck, &configField);
ret = CSC_TransparentCommand(dataToSend, lnIn, &status, &lnOut, bufOut);
if ((ret == RCSC_Ok) && (status == 0x01)) return SUCCESS;
```

CSC_ISOCommand

Description: Sends and receives an ISO 7816 command from the card through the antenna.

Syntax: DWORD WINAPI ***CSC_ISOCommand*** (BYTE* **BufIN**,
DWORD **LnIN**,
BYTE* **BufOUT**,
LPDWORD **lpLnOUT**) ;

Parameters:

I	BYTE*	**BufIN**	Command to send
I	DWORD	**LnIN**	Length of the command to send
O	BYTE*	**BufOUT**	Received answer
O	LPDWORD	**lpLnOUT**	Length of the received answer

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_SendReceive

Example:

```
DWORD     ret;
DWORD     LnIn=7;
DWORD     LnOut;
BYTE      BufIn[7] = {0x94,0xA4,0x00,0x00,0x02,0x3F,0x00};
BYTE      BufOut[LN_OUT_MAX];

ret=CSC_ISOCommand(BufIn,LnIn,BufOut, &LnOut);
```

CSC_ISOCommandContact

Description: Sends and receives a ISO 7816 command from the card in contact mode.
The command can be sent in the 3 mode cases : IN, OUT, IN and OUT.
A 'Select SAM' in ISO7816 protocol on the right slot should have been performed before.

Syntax: DWORD WINAPI ***CSC_ISOCommandContact*** (BYTE* **BufIN**,
DWORD **LnIN**,
BYTE **Case**,
BYTE* **BufOUT**,
DWORD* **lpLnOUT**) ;

Parameters:

I	BYTE*	**BufIN**	ISO Command to send to the card
I	DWORD	**LnIN**	Length of the command to send
I	BYTE	**Case**	ISO 7816 Case :01 : IN 02 : OUT 03 : IN and OUT
O	BYTE*	**BufOUT**	Received answer
O	DWORD*	**lpLnOUT**	Length of the received answer

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_SendReceive

Example:

```
DWORD       ret;
DWORD       LnIn=7;
DWORD       LnOut;
BYTE        BufIn[LnIn] = {0x94,0xA4,0x00,0x00,0x02,0x3F,0x00};
BYTE        BufOut[LN_OUT_MAX];
ret=CSC_SelectSAM(CONTACT_SLOT,1);      // select contact slot in ISO7816 protocol
ret=CSC_ISOCommandContact(BufIn,LnIn,03,BufOut, &LnOut);
ret=CSC_SelectSAM(SAM_SLOT_1,0);        // restore good SAM
```

CSC_SelectSAM

Description : Selects a SAM.
The next operations requiring the SAM will use the selected SAM (for example, GTML_UpdateRecord).

Syntaxe : DWORD WINAPI ***CSC_SelectSAM*** (BYTE **N_SAM,**
BYTE **Type**) ;

Paramètres :

I	BYTE	**N_SAM**	SAM number / slot to select : SAM_CURRENT SAM_SLOT_1 SAM_SLOT_2 SAM_SLOT_3 SAM_SLOT_4 CONTACT_SLOT
I	BYTE	**Type**	Protocol type to use : - 0x00 HSP Innovatron protocol. - 0x01 ISO 7816 protocol.

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_ISOCommandSAM, CSC_ResetSAM

Example:

```
DWORD       ret;

ret=CSC_SelectSAM(SAM_SLOT_1,0);          // In HSP protocol
```

CSC_ResetSAM

Description: Initializes the SAM.

Syntax: DWORD WINAPI ***CSC_ResetSAM*** (BYTE* **lpATR**,
LPDWORD **lpcbATR**) ;

Parameters:

o	BYTE*	**lpATR**	ATR value
o	LPDWORD	**lpcbATR**	Length of the ATR value

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_ISOCommandSAM

Example:

```
DWORD     ret;
BYTE      atr[ATR_LENGTH_MAX];
DWORD     atrLength;

ret=CSC_ResetSAM(atr,&atrLength);
```

CSC_ResetSAMExt

Description: Reset the SAM, and returns the ATR.

Syntax: DWORD WINAPI ***CSC_ResetSAMExt*** (BYTE **SamNum**,
BYTE **SelectINN**,
BYTE **SelectISO**,
LPDWORD **lpcbATR**,
BYTE* **lpATR**);

Parameters:

I	BYTE	**SamNum**	Selection of SAM (1 byte) $00 : SAM usually selected $01 : SAM 1 $02 : SAM 2 $03 : SAM 3 $04 : SAM 4
I	BYTE	**SelectINN**	Selection of SAM in Innovatron High Speed protocol (1 byte) $01 : selection of SAM in Innovatron protocol $00 : no SAM selection in this protocol
I	BYTE	**SelectISO**	selection of protocol ISO 7816 (1 byte) $01 : selection of SAM in ISO7816 T=0 protocol $02 : selection of SAM in ISO7816 T=1 protocol $00 : no SAM selection in this protocol
O	BYTE*	**lpATR**	Contains the ATR of the SAM (n byte)
O	LPDWORD	**lpcbATR**	Length of the ATR value

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

CSC_ISOCommandSAM

Description: Sends and receives a ISO 7816 command to the currently selected SAM, using the HSP protocol.

Syntax:

```
DWORD WINAPI CSC_ISOCommandSAM ( BYTE* BufIN,
                                 DWORD LnIN,
                                 BYTE* BufOUT,
                                 LPDWORD lpLnOUT ) ;
```

Parameters:

I	BYTE*	**BufIN**	Command to send
I	DWORD	**LnIN**	Length of the command to send
O	BYTE*	**BufOUT**	Received answer
O	LPDWORD	**lpLnOUT**	Length of the received answer

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CSC_ResetSAM

Example:

```
DWORD       ret;
DWORD       LnIn=9;
DWORD       LnOut;
BYTE        BufIn[9] = {0x94, 0x14, 0x00, 0x00, 0x04, 0x11, 0x22, 0x33, 0x44};
BYTE        BufOut[LN_OUT_MAX];

ret=CSC_SelectSAM(SAM_SLOT_1,0);          // select SAM in HSP protocol
ret=CSC_ISOCommandSAM(BufIn,LnIn,BufOut, &LnOut);
```

CSC_ISOCommandSAMExt

Description: Sends and receives a ISO 7816 command to the currently selected SAM, using the HSP protocol.

Syntax: DWORD WINAPI ***CSC_ISOCommandSAMExt***(BYTE **NumSAM**,
DWORD **LgBufIN**,
BYTE* **BufIN**,
BYTE **Direction**,
LPDWORD **LgBufOUT**,
BYTE* **BufOUT**) ;

Parameters:

I	BYTE	**NumSAM**	Sam Number $00, $01, $02, $03, $04 as defined in "Reset Sam" cmd
I	DWORD	**LgBufIN**	Length of the command to send
I	BYTE*	**BufIN**	Command to send
I	DWORD	**Direction**	Direction : $01 : In $02 : Out $03 : In - Out
O	BYTE*	**LgBufOUT**	Length of the received answer
O	LPDWORD	**BufOUT**	Received answer

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

CSC_SetSAMBaudratePPS

Description: Perform a PPS on SAM using ISO17816 mode.

Syntax: DWORD WINAPI ***CSC_SetSAMBaudratePPS***(BYTE **ProProt**, BYTE **ParamFD**, WORD ***Status)**

Parameters:

I	BYTE	**ProProt**	Proposed protocol (0 for T=0; 1 for T=1)
I	BYTE	**ParamFD**	FiDi parameter
O	WORD*	**Status**	$0000 : OK $FFFF: error on 1st received byte $FFFE: error on 2nd received byte $FFFD: error on 3rd received byte $FFFC: error on 4th received byte

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

CSC_AntennaOFF

Description: Stops the antenna electromagnetic field.

Syntax: DWORD WINAPI ***CSC_AntennaOFF*** (void) ;

Parameters: None

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	Transmission error
RCSC_CheckSum	CRC Error

See also: CSC_CardStopSearch, CSC_CardEnd

Example:

```
DWORD        ret;
ret=CSC_AntennaOFF();
```

CSC_Switch_Led_Buz

Description : Activates or deactivates the interface signals (Led or Buzzer).

Syntax : DWORD WINAPI ***CSC_Switch_Led_Buz*** (WORD **Param**) ;

Parameter :

I	WORD	**Param**	Word of 16 bits: CPU description: Bit 0 : LED 1 activ if 1 (CSC_CPU_LED1). Bit 1 : LED 2 activ if 1 (CSC_CPU_LED2). Bit 2 : LED 3 activ if 1 (CSC_CPU_LED3). antenna description: Bit 0 : Buzzer activ if 1 (CSC_ANT_BUZZER). Bit 1 : LED 1 activ if 1 (CSC_ANT_LED1). Bit 2 : LED 2 activ if 1 (CSC_ANT_LED2).

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	Transmission error
RCSC_CheckSum	CRC Error

Example:

```
DWORD        ret;
ret=CSC_Switch_Led_Buz (CSC_CPU_LED1| CSC_CPU_LED2 | CSC_ANT_LED2);
```

CSC_SelectCID

Description : Change the currently selected card communication channel in case of multi-activation of cards (ISO14443AorB).

Syntax : DWORD WINAPI ***CSC_SelectCID***(BYTE **CID**, BYTE ***Status**);

Parameter :

I	BYTE	**CID**	Index from 1 to 15 of ISO14443 Card communication channel
O	BYTE*	**Status**	Status of operation 1 = Ok, 0 = Nok (Bad CID value)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	Transmission error
RCSC_CheckSum	CRC Error
RCSC_InputDataWrong	Bad input parameter

Example:

```
DWORD       ret;
BYTE        Status;
ret = CSC_SelectCID(1, &Status);
```

CSC_SelectDIV

Description : Initiate the cryptographic algorithm in the case of Multi SAM usage.

Syntax : DWORD WINAPI ***CSC_SelectDIV***(BYTE **Slot**,
BYTE **Prot**,
BYTE ***DIV**,
BYTE ***Status**);

Parameter :

I	BYTE	**Slot**	Slot of the Secure Access Module
I	BYTE	**Prot**	0 for Innovatron HSP protocol, 1 for ISO7816 protocol
I	BYTE*	**DIV**	4 bytes number used for algorithm diversification
O	BYTE*	**Status**	Status of operation 1 = Ok, 0 = Nok

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error
RCSC_InputDataWrong	Bad input parameter

Example:

```
DWORD       ret;
BYTE        Status;
BYTE        DIV[4] = DIVERSIFICATION_ALG;

ret = CSC_SelectDIV(1, 0, DIV, &Status);
```

CSC_EHP_PARAMS

Description : Change the currently selected card communication channel in case of multi-activation of cards (ISO14443AorB).

Syntax : DWORD WINAPI ***CSC_EHP_PARAMS***(BYTE **MaxNbCard**,
BYTE **Req**,
BYTE **NbSlot**,
BYTE **AFI**
BYTE **AutoSelDiv**);

Parameter :

I	BYTE	**MaxNbCard**	Max number of card to look for (also limited in CSC) this value activate or de-activate the multi-card search (default value is 0x01)
I	BYTE	**Req**	0 for REQ, 1 for WakeUp (default value is 0x01)
I	BYTE	**NbSlot**	Number of slot required for anticollision (RFU:should stay to 0)
I	BYTE	**AFI**	AFI value for anticollision (default value is All = 0x00)
I	BYTE	**AutoSelDiv**	Automatic selection of diversifier with the current security access module (default value is yes = 0x01)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error
RCSC_InputDataWrong	Bad input parameter

Example:

```
DWORD       ret;

ret = CSC_EHP_PARAMS(5, 1, 0, 0, 0);
```

CSC_EHP_PARAMS_EXT

Description : Change the currently selected card communication channel in case of multi-activation of cards (ISO14443AorB).

Syntax :

DWORD WINAPI ***CSC_EHP_PARAMS_EXT***(BYTE **MaxNbCard**,
BYTE **Req**,
BYTE **NbSlot**,
BYTE **AFI**
BYTE **AutoSelDiv,**
BYTE **Deselect,**
BYTE **SelectAppli,**
BYTE **Lg,**
LPBYTE **Data,**
WORD **FelicaAFI,**
BYTE **FelicaNbSlot**);

Parameter :

I	BYTE	**MaxNbCard**	Max number of card to look for (also limited in CSC) this value activate or de-activate the multi-card search (default value is 0x01)
I	BYTE	**Req**	0 for REQ, 1 for WakeUp (default value is 0x01)
I	BYTE	**NbSlot**	Number of slot required for anticollision (RFU:should stay to 0)
I	BYTE	**AFI**	AFI value for anticollision (default value is All = 0x00)
I	BYTE	**AutoSelDiv**	Automatic selection of diversifier with the current security access module (default value is yes = 0x01)
I	BYTE	**Deselect**	0 switch field off / 1 real deselection of the found cards (1 byte)
I	BYTE	**SelectAppli**	$000xxxx1 : send select appli to card after detection (1 byte) $000xxx1x : force to $00 (instead of $94) the select appli "CLA" field $000x1xxx : add selected appli name in the EnterHuntPhase answer
I	BYTE	**Lg**	Optional data Length "n" (1 byte)
I	BYTE	**Data**	Optional name of the appli to select(default value is "1TIC") (n byte)
I	BYTE	**FelicaAFI**	Card function identifier (default is all cards = $FFFF) (2 byte)
I	BYTE	**FelicaNbSlot**	Slot Number for Felica Anticollision (default value = 3) (1 byte)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error
RCSC_InputDataWrong	Bad input parameter

CSC_WriteSAMNumber

Description : Writes the default SAM number in EEPROM for memory.

Syntax : DWORD ***CSC_WriteSAMNumber***(BYTE ***N_SAM***,
BYTE* ***status***);

Parameter :

I	BYTE	**N_SAM**	SAM number : SAM_SLOT_1 SAM_SLOT_2 SAM_SLOT_3 SAM_SLOT_4 CONTACT_SLOT
O	BYTE*	**status**	Execution status : $00 : Failure $01 : Success

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error
RCSC_InputDataWrong	Bad input parameter

Example:

```
DWORD       ret;
BYTE        status;

ret = CSC_WriteSAMNumber(SAM_SLOT_1, &status);
```

1.8. EMVCo functions

These functions allow access to GEN5XX EMVCo features. For more information, see “EMV MANAGEMENT” in the “RD-ST-08167-xx_ASK CSC - Coupler Software Interface_Gen5XX.pdf” document.

EMVCo_UserInterface

Description: Performs EMV’s standard LEDs activation and buzzer tones.

Syntax: DWORD WINAPI ***EMVCo_UserInterface*** (BYTE **SequenceNumber,**
LPBYTE **Status**);

Parameters:

I	BYTE	**SequenceNumber**	$01: Not ready: all LEDs off, buzzer off $02: Idle: LED1 on during 200 ms, buzzer off $03: Ready to Read: LED1 on, buzzer off Option 1 for Card Read Successfully / Processing Error $11: Card read successfully: 4 LEDs in sequence (250 ms sequence and 750 ms remaining) and success tone $12: Processing Error: all LEDs off and alert tone Option 2 for Card Read Successfully / Processing Error $21: Card read successfully: 3 LEDs in sequence (125 ms sequence and 750 ms remaining) and success tone. $22: Processing Error: LED 4 on and alert tone
O	LPBYTE	**Status**	$00: Ok $FF: Sequence not implemented

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error

Example:

```
DWORD       dwRet;
BYTE        bStatus;
dwRet=EMVCo_UserInterface(0x11,& bStatus); // visual and audible sequence indication successful payment
```

EMVCo_Contactless

Description: Performs EMV's featured commands: RF field reset, PICC activation or PICC removal.

Syntax: DWORD WINAPI ***EMVCo_Contactless*** (BYTE **CommandNumber,**
LPBYTE **Parameters,**
LPBYTE **Status,**
LPBYTE **Length,**
LPBYTE **PICCData**);

Parameters:

I	BYTE	**CommandNumber**	$00: RF field off, Parameters empty $01: RF field reset, Parameters empty $02: Polling / Anti-collision / Activation, Parameters, 1 byte = number of polling loops $03: Removal, Parameters, 1 byte = number of polling loops $04: EMV internal loop-back, Parameters, 1 byte = number of loops ($FF = infinite) $05: Set/Reset EMV flag. Parameters, 1 byte = EMV flag value. This allows to set/reset the EMV behavior (EMV's ISO14443 implementation), using no EMV command set. This flag is automatically managed if EMV command set is used. $06: Polling / Anti-collision / Activation + other technologies polling. Parameters: Byte 1: number of polling loops Byte 2 to byte 6: EHP parameters (5 first bytes), as define in "Enter Hunt Phase" command.
I	LPBYTE	**Parameters**	See above
O	LPBYTE	**Status**	$00: No PICC found or action accomplished $01: Type A PICC found $02: Type B PICC found $03: Type INNOVATRON PICC found $06: Type CTS/CTM PICC found $0D: Type ST SR PICC found $0E: Type Felica PICC found $10: More than one PICC found $11: Communication error $12: Timeout error $FF: Command not implemented
I O	LPBYTE	**Length**	Length of the PICC data

o	LPBYTE	**PICCData**	PICC data depends of the PICC type detected: Type A: ATQA (2), SAK (1), UID Length (1), UID (UID Length), ATS (ATS Length) Type B: ATQB (12), ATTRIB Response Length (1), ATTRIB Response (ATTRIB Response Length) Other types: no data.

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error

Note:

After EMV card detection, card commands are sent using CSC_ISOCommand () function.

Example:

```
DWORD       dwRet;
BYTE        bStatus;
BYTE        Parameters[1];
BYTE        Length;
BYTE        PICCData[50];
DWORD       LnOut;
BYTE        BufOut[LN_OUT_MAX];

Parameters[0] = 16;             // 16 polling loops
dwRet = EMVCo_Contactless (0x02, Parameters, &bStatus, &Length, PICCData);      // detect card

if (dwRet == RCSC_Ok) && ((bStatus == 0x01) || (bStatus == 0x02)))    // ISOA or ISOB detected
{       // select application                   // 2 P A Y . S Y S . D D F 0 1
        dwRet=CSC_ISOCommand("\x00\xA4\x04\x00\x0E\x32\x50\x41\x59\x2E\x53\x59\x53\x2E\x44\x44\x46\x3
        0\x31\x00",20,BufOut, &LnOut);
}
dwRet = EMVCo_Contactless (0x03, Parameters, &bStatus, NULL, NULL);      // card removal
```

1.8. Calypso Rev3 functions

These functions allow access to Calypso Rev3 features. For more information, see “CALYPSO REV3 MANAGEMENT” in the “RD-ST-08167-xx_ASK CSC - Coupler Software Interface_Gen5XX.pdf” document.

Others functions has been added to cover Calypso Rev3 functionalities, such as ReadRecordMultiple, ReadBinary, UpdateBinary, WriteBinary, SelectApplicationCalypsoRev3 (see “1.11. Variable class mapping”).

For all of these functions the error code table is listed here-under:

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The received data are wrong
RCSC_CheckSum	CRC Error

CalypsoRev3_GetMode

Description: Get the Calypso Rev3 mode flag. In Calypso Rev3 mode, the reader will try to manage the card as a Calypso rev3 card if the card is compliant.

Syntax: DWORD WINAPI ***CalypsoRev3_GetMode*** (BYTE ***Mode**);

Parameters:

O	BYTE	***Mode**	$00: Calypso Rev3 mode disabled. $01: Calypso Rev3 mode enabled.

CalypsoRev3_SetMode

Description: Set the Calypso Rev3 mode flag. In Calypso Rev3 mode, the reader will try to manage the card as a Calypso rev3 card if the card is compliant.

Syntax: DWORD WINAPI ***CalypsoRev3_SetMode*** (BYTE **Mode**);

Parameters:

I	BYTE	**Mode**	$00: Disable Calypso Rev3 mode. $01: Enable Calypso Rev3 mode.

1.9. GTML card management functions

GTML_SelectFile

Description: Explicit selection of the current EF or DF.

Syntax: DWORD ***GTML_SelectFile*** (BYTE **SelectMode**, LPBYTE **IdPath**, BYTE **IdPathLen**, LPBYTE **FCI**, sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**SelectMode**	Selection control GTML_SEL_MF = MF GTML_SEL_CURENT_EF = EF in the current DF GTML_SEL_PATH = Path from MF (excluded)
I	LPBYTE	**IdPath**	Path or Identifier Ex. '2000' = DF 0x2000 or '20002010' = file 0x2010 in DF 0x2000
I	BYTE	**IdPathLen**	Length of the 'IdPath' path
O	LPBYTE	**FCI**	File description data (Refer to section 6.3.2. Structure of the data in the document ASK v3.0 Coupler User Interface)
O	sCARD _Status*	**Status**	Order execution report

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

GTML_Invalidate

Description: Invalidation of the current DF

Syntax: DWORD ***GTML_Invalidate*** (BYTE **AccMode**, sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See 6.3.2. Data structure)
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_SelectFile, GTML_Rehabilitate

GTML_Rehabilitate

Description: Cancels the invalidation of the current DF

Syntax: DWORD ***GTML_Rehabilitate*** (BYTE **AccMode**, sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_SelectFile, GTML_Invalidate

GTML_ChangePIN

Description: Changes the card's PIN

Syntax: DWORD ***GTML_ChangePIN*** (LPBYTE **OldPIN**,
LPBYTE **NewPIN**,
sCARD_Status* **Status**) ;

Parameters:

I	LPBYTE	**OldPIN**	Value of the old PIN (4 characters)
I	LPBYTE	**NewPIN**	Value of the new PIN (4 characters)
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_SelectFile, GTML_ChangeKey

GTML_VerifyPIN

Description: Verifies the PIN code of the card.

Syntax: DWORD ***GTML_VerifyPIN*** (LPBYTE **PIN**,
sCARD_Status* **Status**) ;

Parameters:

I	LPBYTE	**PIN**	Value of the PIN (4 characters)
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_SelectFile, GTML_ChangePIN

GTML_Increase

Description: Increases the value contained in a coupler file

Syntax: DWORD ***GTML_Increase*** (BYTE **AccMode**,
BYTE **SID**,
DWORD **Value**,
LPDWORD **NewValue**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)
I	BYTE	**SID**	Short file number SID = 00 for the currently selected EF
I	DWORD	**Value**	Value to be increased (24 bits)
O	LPDWORD	**NewValue**	New coupler value (in out of session mode)
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_Decrease

GTML_Decrease

Description: Decreases the value contained in a coupler file

Syntax: DWORD ***GTML_Decrease*** (BYTE **AccMode**,
BYTE **SID**,
DWORD **Value**,
LPDWORD **NewValue**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)
I	BYTE	**SID**	Short file number SID = 00 for the currently selected EF
I	DWORD	**Value**	Value to be deducted (24 bits)
O	LPDWORD	**NewValue**	New coupler value (in out of session mode)
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_Increase

GTML_ReadRecord

Description: Reads the record provided in the EF

Syntax: DWORD ***GTML_ReadRecord*** (BYTE **AccMode**,
BYTE **SID**,
BYTE **NuRec**,
BYTE **DataLen**,
LPBYTE **Data**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)
I	BYTE	**SID**	Short file number SID = 00 for the currrently selected EF
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of the data to be read
O	LPBYTE	**Data**	Data read
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_AppendRecord, GTML_UpdateRecord, GTML_SelectFile

GTML_AppendRecord

Description: Adds a record to a circular EF

Syntax: DWORD ***GTML_AppendRecord*** (BYTE **AccMode**,
BYTE **SID**,
LPBYTE **Rec**,
BYTE **RecSize**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)
I	BYTE	**SID**	Short file number SID = 00 for the currently selected EF
I	LPBYTE	**Rec**	New record data
I	BYTE	**RecSize**	New record data length
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_ReadRecord, GTML_UpdateRecord, GTML_SelectFile

GTML_UpdateRecord

Description: Deletes and writes a record in an EF

Syntax: DWORD ***GTML_UpdateRecord*** (BYTE **AccMode**,
BYTE **SID**,
BYTE **NuRec**,
BYTE **DataLen**,
LPBYTE **Data**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See section 5.3.2. Data Structure)
I	BYTE	**SID**	Short file number SID = 00 for the currently selected EF
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of the data to be written
I	LPBYTE	**Data**	Data to be written
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_ReadRecord,GTML_AppendRecord,GTML_SelectFile, GTML_WriteRecord

GTML_WriteRecord

Description: Writes a record in an EF

Syntax: DWORD ***GTML_WriteRecord*** (BYTE **AccMode**,
BYTE **SID**,
BYTE **NuRec**,
BYTE **DataLen**,
LPBYTE **Data**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)
I	BYTE	**SID**	Short file number SID = 00 for the currently selected EF
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of the data to be written
I	LPBYTE	**Data**	Data to be written
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_ReadRecord, GTML_AppendRecord, GTML_UpdateRecord

GTML_OpenSession

Description: Opens a secured session

Syntax: DWORD ***GTML_OpenSession*** (BYTE **Type**,
BYTE **SID**,
BYTE **NRec**,
sCARD_Session* **Session**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**Type**	Type of operation: - Customization (0x00) - Loading (0x01) - Validation (0x02)
I	BYTE	**SID**	Short file number SID = 00 for the currently selected EF
I	BYTE	**Nrec**	Record number
O	sCARD _Session*	**Session**	Value of data return on applications
O	sCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_CloseSession

GTML_CloseSession

Description: Closes a secured session

Syntax: DWORD ***GTML_CloseSession*** (LPBYTE **Result**,
LPDWORD **cbResult**,
sCARD_Status* **Status**) ;

Parameters:

o	LPBYTE	**Result**	Result from the incoming and outgoing orders
o	LPDWORD	**cbResult**	Result length
o	SCARD _Status*	**Status**	Order execution report (See section 6.3.2. Data Structure in document ASK v3.0 Coupler User Interface)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_OpenSession

GTML_AbortSecuredSession

Description: Stop the current certification session. This still allow to continue a dialogue with the badge and, in particular, open a new session

Syntax: DWORD ***GTML_AbortSecuredSession*** (sCARD_Status* **Status**);

Parameters:

o	SCARD_Status	***Status**	Contains the card execution return status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: GTML_OpenSession

1.10. CD97 card management functions

CD97_SelectFile

Description: Explicit selection of current EF or DF.

Syntax: DWORD ***CD97_SelectFile*** (BYTE **SelectMode**, LPBYTE **IdPath**,
BYTE **IdPathLen**, LPBYTE **FCI**,
sCARD_Status * **Status**) ;

Parameters:

I	BYTE	**SelectMode**	Selection Control CD97_SEL_MF = MF CD97_SEL_CURENT_EF = EF in the current DF CD97_SEL_PATH = Path from MF (excluded)
I	LPBYTE	**IdPath**	Identifier or path Ex. '1000' = DF 0x1000 or '31003115' = file 0x3115 in DF 0x3100
I	BYTE	**IdPathLen**	Length of identifier or path
O	LPBYTE	**FCI**	File description data (see the coupler specification.)
O	sCARD _Status*	**Status**	Execution status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_TXError	transmission error
RCSC_CheckSum	CRC Error

See also: CD97_StatusFile

CD97_StatusFile

Description: Same CD97_SelectFile but without selecting any file.

CD97_Invalidate

Description: Invalidates selected DF (and all children files)

Syntax: DWORD ***CD97_Invalidate*** (BYTE **AccMode**, sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access Mode
O	sCARD _Status*	**Status**	Execution status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_SelectFile, CD97_Rehabilitate

CD97_Rehabilitate

Description: Cancels a file's invalidation

Syntax: DWORD ***CD97_Rehabilitate*** (BYTE **AccMode**, sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access Mode
O	sCARD _Status*	**Status**	Execution status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_SelectFile, CD97_Invalidate

CD97_ChangeKey

Description: Modifies a key

Syntax: DWORD ***CD97_ChangeKey*** (BYTE **KeyIndex**,
BYTE **NewVersion**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**KeyIndex**	Key Index to modify (01..03)
I	BYTE	**NewVersion**	New version of the key (<>00)
O	sCARD _Status*	**Status**	Execution status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_ChangePIN

CD97_ChangeKeyExt

Description: Change the key / Personnalization

Syntax: DWORD ***CD97_ChangeKeyExt*** (BYTE **KeyIndex**,
BYTE **NewKeyVersion**,
BYTE **TypeCmd**,
BYTE **KeyIndexEncipher**,
BYTE **ALGTag**,
BYTE **ALGSam**,
BYTE **NewKeyIndex**,
sCARD_Status* **Status**);

Parameters:

I	BYTE	**KeyIndex**	Key Index to modify (01..03)
I	BYTE	**NewKeyVersion**	New version of the key (<>00)
I	BYTE	**TypeCmd**	type Command (1 byte) $00 : short cmd $01 : long cmd
I	BYTE	**KeyIndexEncipher**	Index of the key to encipher the transfer (1 byte)
I	BYTE	**ALGTag**	Algo key card to recopy (1 byte)
I	BYTE	**ALGSam**	Algo of the Sam used (1 byte)
I	BYTE	**NewKeyIndex**	index of the new key in the card in the DF (1 byte)
O	sCARD_Status	***Status**	Contains the card execution return status (3 bytes)

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_ChangePIN

CD97_ChangePIN

Description: Changes the PIN code of the card

Syntax: DWORD ***CD97_ChangePIN*** (LPBYTE **OldPIN**,
LPBYTE **NewPIN**,
sCARD_Status* **Status**) ;

Parameters:

I	LPBYTE	**OldPIN**	Old PIN code value (4 characters)
I	LPBYTE	**NewPIN**	New PIN code value (4 characters)
O	sCARD _Status*	**Status**	Execution status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_SelectFile, CD97_ChangeKey

CD97_ChangePINExt

Description: Changes the PIN code of the card

Syntax: DWORD ***CD97_ChangePINExt*** (BYTE **KeyNum**,
LPBYTE **OldPIN**,
LPBYTE **NewPIN**,
BYTE **TypeCmd**,
BYTE **KeyNumKIF**,
BYTE **KVC**,
BYTE **ALG**,
BYTE **SamNum**,
sCARD_Status* **Status**);

Parameters:

I	LPBYTE	**KeyNum**	Key number (1 byte) $00 : CD97, GTML and CT2000, $04 : GTML2 and CD21, $09 : POPEYE
I	LPBYTE	**OldPIN**	Old PIN Code (4 bytes)
I	LPBYTE	**NewPIN**	New PIN Code (4 bytes)
I	LPBYTE	**TypeCmd**	type Command (1 byte) $00 : short cmd $01 : long cmd
I	LPBYTE	**KeyNumKIF**	SAM key number to use (1 byte) or KIF of the key
I	LPBYTE	**KVC**	$00 (if NKEY passed in the previous parameter)(1 byte) or KVC of the Key
I	LPBYTE	**ALG**	Algorithm of the SAM used (1 byte)
I	LPBYTE	**SamNum**	SAM number (1 byte) $00 : default SAM, $01, $02, $03 or $04 : logical number of the wanted SAM number
O	sCARD_Status	***Status**	Contains the card execution return status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

CD97_VerifyPIN

Description: Verifies the PIN code of the card

Syntax: DWORD ***CD97_VerifyPIN*** (LPBYTE **PIN**,
sCARD_Status* **Status**) ;

Parameters:

I	LPBYTE	**PIN**	PIN code value (4 characters)
O	sCARD _Status*	**Status**	Execution status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_SelectFile, CD97_ChangePIN

CD97_VerifyPINExt

Description: Verifies the PIN code of the card

Syntax: DWORD ***CD97_VerifyPINExt*** (BYTE Mode,
LPBYTE PIN,
BYTE TypeCmd,
BYTE KeyNumKIF,
BYTE KVC,
BYTE SamNum,
sCARD_Status* Status);

Parameters:

I	BYTE	**Mode**	Mode (1 byte) $00 : consultation of counter of number of incorrect presentations $01 : presentation of PIN $02 : presentation of PIN in transparent mode for contact communication
I	LPBYTE	**PIN**	PIN code (4 bytes)
I	BYTE	**TypeCmd**	Type Cmd (1 byte) $00 : short command (compatibility with the former one) $01 : long command
I	BYTE	**KeyNumKIF**	SAM key number to use Or KIF of the key (1 byte)
I	BYTE	**KVC**	$00 if NKEY passed in the previous parameter or KVC of the Key (1 byte)
I	BYTE	**SamNum**	SAM number (1 byte) $00 : default SAM, $01, $02, $03 or $04 : logical number of the wanted SAM number
O	sCARD_Status	***Status**	Contains the card execution return status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_SelectFile, CD97_ChangePIN

CD97_Increase

Description: Increases value contained in a counter file

Syntax: DWORD ***CD97_Increase*** (BYTE **AccMode**,
BYTE **SID**,
DWORD **Value**,
LPDWORD **NewValue**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode
I	BYTE	**SID**	Small identifier file number SID = 00 for EF usually selected
I	DWORD	**Value**	Value to add (24 bits)
O	LPDWORD	**NewValue**	New value of the counter
O	sCARD _Status*	**Status**	Execution status

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_Decrease

CD97_Decrease

Description: Decreases value contained in counter's file

Syntax: DWORD ***CD97_Decrease*** (BYTE **AccMode**,
BYTE **SID**,
DWORD **Value**,
LPDWORD **NewValue**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode
I	BYTE	**SID**	Small identifier file number SID = 00 for EF usually selected
I	DWORD	**Value**	Value to deduce (24 bits)
O	LPDWORD	**NewValue**	New value of the counter
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_Increase

CD97_ReadRecord

Description: Reads record in circular or linear EF

Syntax: DWORD ***CD97_ReadRecord*** (BYTE **AccMode**,
BYTE **SID**,
BYTE **NuRec**,
BYTE **DataLen**,
LPBYTE **Data**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode
I	BYTE	**SID**	Small identifier file number SID = 00 for EF usually selected
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of data to be read
O	LPBYTE	**Data**	read data
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_AppendRecord, CD97_UpdateRecord, CD97_SelectFile

CD97_AppendRecord

Description: Adds a record to circular EF

Syntax: DWORD ***CD97_AppendRecord*** (BYTE **AccMode**,
BYTE **SID**,
LPBYTE **Rec**,
BYTE **RecSize**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode
I	BYTE	**SID**	Small identifier file number SID = 00 for EF usually selected
I	LPBYTE	**Rec**	New record data
I	BYTE	**RecSize**	Length of data
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_ReadRecord, CD97_UpdateRecord, CD97_SelectFile

CD97_UpdateRecord

Description: Erases and writes a record in an EF file

Syntax: DWORD ***CD97_UpdateRecord*** (BYTE **AccMode**,
BYTE **SID**,
BYTE **NuRec**,
BYTE **DataLen**,
LPBYTE **Data**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode
I	BYTE	**SID**	Small identifier file number SID = 00 for EF usually selected
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of data to write
I	LPBYTE	**Data**	Data to write
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_ReadRecord,CD97_AppendRecord,CD97_SelectFile, CD97_WriteRecord

CD97_WriteRecord

Description: Writes a record in an EF file

Syntax: DWORD ***CD97_WriteRecord*** (BYTE **AccMode**,
BYTE **SID**,
BYTE **NuRec**,
BYTE **DataLen**,
LPBYTE **Data**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**AccMode**	Access mode
I	BYTE	**SID**	Small identifier file number SID = 00 for EF usually selected
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of data to write
I	LPBYTE	**Data**	Data to write
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_ReadRecord, CD97_AppendRecord, CD97_UpdateRecord

CD97_OpenSession

Description: Secure session opening

Syntax: DWORD ***CD97_OpenSession*** (BYTE **Type**,
BYTE **SID**,
BYTE **NRec**,
sCARD_Session* **Session**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**Type**	Operation type: - Personalization (0x00) - Reload (0x01) - Validation (0x02)
I	BYTE	**SID**	Small identifier file number SID = 00 for EF usually selected
I	BYTE	**Nrec**	Record number
O	sCARD _Session*	**Session**	Application data return value
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_CloseSession

CD97_ OpenSessionExt

Description: Secure session opening

Syntax: DWORD ***CD97_ OpenSessionExt*** (BYTE **Type**,
sCARD_SecurParam **Secur**,
BYTE **RecNum**,
BYTE **TypeCmd**,
BYTE **Mode**,
sCARD_Status* **Status**,
sCARD_Session* **Session**,
BYTE* **KVC**);

Parameters:

I	BYTE	**Type**	Operation type: - Personalization (0x00) - Reload (0x01) - Validation (0x02)
I	sCARD_SecurParam	**Secur**	Contain the parameters for the security SID: Short ID Number (ex: CD97_SID_RT_JOURNAL, ...) NKEY : Number of Key which use in the SAM (in future KIF) RUF : Reserved for KVC
I	BYTE	**RecNum**	Record number
I	BYTE	**TypeCmd**	Type Cmd $00 : short command (compatibility with the former one for CD97 and GTML) $01 : long command
I	BYTE	**Mode**	Mode of operation $00 : simple mode $01 : extended mode
O	sCARD_Status	***Status**	Contains the card execution return status
O	sCARD_Session	***Session**	Contains the application data return value - NbApp - Path[128] - Data[29] - KVC
O	BYTE	***KVC**	KVC in extended mode

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_CloseSession

CD97_CloseSession

Description: Secure session closing

Syntax: DWORD ***CD97_CloseSession*** (LPBYTE **Result**,
LPDWORD **cbResult**,
sCARD_Status* **Status**) ;

Parameters:

o	LPBYTE	**Result**	Order result
o	LPDWORD	**cbResult**	Length of order result
o	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_OpenSession

CD97_CloseSessionExt

Description: Secure session closing

Syntax: DWORD ***CD97_CloseSessionExt*** (BYTE **TypeCmd**,
BYTE **TimeOut**,
sCARD_Status* **Status**,
LPDWORD **LgResult**,
LPBYTE **Result**) ;

Parameters:

I	BYTE	**TypeCmd**	Type Cmd (1 byte) $00 : session will be ratified at the reception of the following command $80 : session is ratified immediately (except for CD97 and GTML) $4A : switches OFF the field if the card doesn't answer
I	BYTE	**TimeOut**	if TYPE=$4A
O	sCARD_Status	***Status**	Contains the card execution return status
O	LPDWORD	**LgResult**	The Result length (1 byte)
O	LPBYTE	**Result**	Order result (n bytes)

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_OpenSession

CD97_AbortSecuredSession

Description: Stop the current certification session. This still allow to continue a dialogue with the badge and, in particular, open a new session

Syntax: DWORD ***CD97_AbortSecuredSession*** (sCARD_Status* **Status**);

Parameters:

o	sCARD_Status	***Status**	Contains the card execution return status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

CD97_SelectISOApplication

Description: Select application using Select File ISO command

Syntax: DWORD ***CD97_SelectISOApplication*** (BYTE SelectOption,
BYTE Lg,
LPBYTE Data,
sCARD_Status* Status,
LPBYTE FCI);

Parameters:

I	BYTE	**SelectOption**	Select Option (1 byte) 00 : first application or select by name if LNG <> 0. 01 : select last application (LNG should be 0) 02 : select next application (LNG should be 0) 03 : select previoust application (LNG should be 0)
I	BYTE	**Lg**	length of data "n" (1 byte) 0 if Select Option <> 0, otherwise <= 16
I	LPBYTE	**Data**	Application Name (n bytes)
O	sCARD_Status	***Status**	Contains the card execution return status
O	LPBYTE	**FCI**	FCI (n bytes)

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

CD97_Purchase

Description: Purchases with EP

Syntax: DWORD ***CD97_Purchase*** (BYTE **Type**,
LPBYTE **DataLog**,
LPBYTE **Disp**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**Type**	Purchase type 00 = purchase transaction 01 = purchase transaction with display
I	LPBYTE	**DataLog**	7 first bytes of payments' log new recording
I	LPBYTE	**Disp**	6 bytes display
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_GetEPStatus

CD97_GetEPStatus

Description: Returns the EP value and prepares for purchasing or reloading

Syntax: DWORD ***CD97_GetEPStatus*** (BYTE **Type**,
LPDWORD **EP**,
LPBYTE **Log**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**Type**	Transaction type to achieve 00 = Reload transaction 01 = Purchase transaction 02 = Cancel purchase
O	LPDWORD	**EP**	EP Value
O	LPBYTE	**Log**	If Type = 00 then Log = Reload log (22 car.) If Type = 01 then Log = Paid log (19 car.) If Type = 02 then Log = Paid log (19 car.)
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_Purchase

CD97_ReloadEP

Description: Reloads EP

Syntax: DWORD ***CD97_ReloadEP*** (LPBYTE **ChargLog1**,
LPBYTE **ChargLog2**,
sCARD_Status* **Status**) ;

Parameters:

I	LPBYTE	**ChargLog1**	5 first bytes of new reload log record: - Date (2 bytes) since 1/1/97 in days - Money batch (2 bytes) - Equipment type (1 byte)
I	LPBYTE	**ChargLog2**	5 bytes, offset [0x08..0x13] new reload log record: - Amount (3 bytes) - Time (2 bytes) Time since 00:00:00 in minutes.
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_GetEPStatus

CD97_CancelPurchase

Description: Cancels the last payment made by EP

Syntax: DWORD ***CD97_CancelPurchase*** (BYTE **Type**,
LPBYTE **DataLog**,
LPBYTE **Disp**,
sCARD_Status* **Status**) ;

Parameters:

I	BYTE	**Type**	Purchase type 00 = purchase transaction 01 = purchase transaction with display
I	LPBYTE	**DataLog**	7 first bytes of payments' log new recording
I	LPBYTE	**Disp**	6 bytes display
O	sCARD _Status*	**Status**	Execution status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: CD97_Purchase

1.11. Variable class mapping

Before the execution of each command you must configure the security parameters of the sCARD_SecurParam structure:

- ✓ AccMode : used mode for the command execution (STAMPED, PROTECTED, DEFAULT)
- ✓ SID : Short ID of the file used in the command.
- ✓ LID : Long ID of the file used in the command.
- ✓ NKEY : Key number to use (in the SAM), in order to execute the command (For future use it will be the KIF).
- ✓ RFU : For future use it will be the KVC (actually 0x00).

SelectFile

Description : Explicit selection of current EF or DF.

Syntax : DWORD ***SelectFile*** (BYTE **SelectMode,**
LPBYTE **IdPath,**
BYTE **IdPathLen,**
LPBYTE **FCI,**
sCARD_Status* **Status**) **;**

Parameters :

I	BYTE	**SelectMode**	Selection control GEN_SEL_MF = MF GEN_SEL_CURENT_EF = EF in the current DF GEN_SEL_PATH = path from MF (excluded)
I	LPBYTE	**IdPath**	Identifier or path Ex. '1000' = DF 0x1000 or '31003115' = file 0x3115 in the DF 0x3100
I	BYTE	**IdPathLen**	Length of identifier or path 'IdPath'
O	LPBYTE	**FCI**	File description data.
O	sCARD _Status*	**Status**	Execution status

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : StatusFile

Example: (select MF)

```
DWORD         ret ;
BYTE          SEL_MF[2]={0x3F,0x00};
BYTE          FCI[FCI_MAX_LENGTH];
sCARD_Status  Status;

ret=SelectFile(GEN_SEL_MF,SEL_MF,sizeof(SEL_MF),FCI,&Status);
```

StatusFile

Description : Same as SelectFile function but without selecting any file.

Example: (Status MF)

```
DWORD         ret ;
BYTE          SEL_MF[2]={0x3F,0x00};
BYTE          FCI[FCI_MAX_LENGTH];
sCARD_Status  Status;

ret=StatusFile(GEN_SEL_MF,SEL_MF,sizeof(SEL_MF),FCI,&Status);
```

Invalidate

Description : Invalidates current DF.

Syntax : DWORD ***Invalidate*** (sCARD_SecurParam **SecurParam,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, LID, NKEY.
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : SelectFile, Rehabilitate

Example : (Invalidate RT in protected mode (CD97))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
BYTE                SEL_RT[2]={0x20,0x00};

Secur.AccMode=GEN_ACCESS_MODE_PROTECTED;
Secur.RFU=0x00;
Secur.SID=0x00;
Secur.LID=0x2000;
Secur.NKEY=RT_TAC_KEY;    // valid

ret=SelectFile(GEN_SEL_PATH,SEL_RT,sizeof(SEL_RT),tx,&Status);
ret=Invalidate(Secur,&Status);
```

Rehabilitate

Description : Cancels the current DF invalidation.

Syntax : DWORD ***Rehabilitate*** (sCARD_SecurParam **SecurParam,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, LID, NKEY.
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : SelectFile, Invalidate

Example : (Rehabilitate RT in session (GTML))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;
BYTE                lnOut;
BYTE                bufOut[LN_OUT_MAX];

Secur.AccMode=GEN_ACCESS_MODE_DEFAULT;
Secur.RFU=0x00;
Secur.SID=0x00;
Secur.LID=0x2000;
Secur.NKEY=RT_UPD_KEY;    // perso

ret=OpenSession(0x00,Secur,00,&Session,&Status);
ret=Rehabilitate(Secur,&Status);
ret=CloseSession(bufOut,&lnOut,&Status);
```

ChangePIN

Description : Changes the PIN code of the card.

Syntax :

DWORD ***ChangePIN*** (sCARD_SecurParam **SecurParam,**
LPBYTE **OldPIN,**
LPBYTE **NewPIN,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: NKEY (Key number to use in the SAM in order to cipher the PIN).
I	LPBYTE	**OldPIN**	Old PIN code value (4 characters)
I	LPBYTE	**NewPIN**	New PIN code value (4 characters)
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : SelectFile, ChangeKey

Example : (CT2000)

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
BYTE                OldPIN[4] = {0x30, 0x30, 0x30, 0x30};
BYTE                NewPIN[4] ={0x31, 0x31, 0x31, 0x31};

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID=0x00;
Secur.LID=0x00;
Secur.NKEY=MF_PER_KEY;

ret=ChangePIN(Secur,OldPIN,NewPIN,&Status);
```

VerifyPIN

Description : Verifies the PIN code of the card.

Syntax : DWORD ***VerifyPIN*** (sCARD_SecurParam **SecurParam,**
LPBYTE **PIN,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: NKEY :Key number to use in the SAM in order to cipher the PIN. If NKEY=0x00, PIN presentation in clear mode. Other parameters : RFU
I	LPBYTE	**PIN**	PIN code value (4 characters)
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : SelectFile, ChangePIN, PINStatus

Example : (CT2000)

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
BYTE                PIN[4] = {0x30, 0x30, 0x30, 0x30};

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID=0x00;
Secur.LID=0x00;
Secur.NKEY=MF_INV_KEY;      // in crypted mode. (0x00 for clear mode).

ret=VerifyPIN(Secur, PIN, &Status);
```

PINStatus

Description : Checks the PIN code status (number of incorrect presentations). A 'SelectFile' on the MF should have been preformed before.

Syntax : DWORD ***PINStatus*** (sCARD_Status* **Status**) **;**

Parameters :

o	sCARD _Status*	**Status**	Execution status : - $ 00.90.00 : no incorrect presentations - $ 02.63.Cx : x represents the number of PIN presentations still authorized

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : SelectFile, ChangePIN, VerifyPIN

Example :

```
DWORD          ret;
sCARD_Status   Status;

ret=PINStatus(&Status);
```

Increase

Description : Increases value contained in a counter file.

Syntax :

```
DWORD Increase ( sCARD_SecurParam SecurParam,
                 BYTE ICount,
                 DWORD Value,
                 LPDWORD NewValue,
                 sCARD_Status* Status ) ;
```

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**ICount**	Index of the counter to be incremented
I	DWORD	**Value**	Value to add (24 bits)
O	LPDWORD	**NewValue**	New value of the counter (depend of the card and mode)
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: Decrease

Example: (Decrease counter 1 in reload session (GTML))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;
DWORD               value=0x101010;
DWORD               newValue
BYTE                lnOut;
BYTE                bufOut[LN_OUT_MAX];

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID= GTML_SID_RT_EVENTS_LOG;
Secur.LID=0x00;
Secur.NKEY=RT_RLD_KEY;

ret=OpenSession(01,Secur,01,&Session,&Status);
Secur.SID=GTML_SID_RT_COUNTER_1;
ret=Increase(Secur,0,value,&newValue,&Status);
ret=CloseSession(bufOut,&lnOut,&Status);
```

Decrease

Description : Decreases value contained in counter's file.

Syntax :

```
DWORD Decrease ( sCARD_SecurParam SecurParam,
                 BYTE ICount,
                 DWORD Value,
                 LPDWORD NewValue,
                 sCARD_Status* Status ) ;
```

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**ICount**	Index of the counter to be decremented
I	DWORD	**Value**	Value to deduce (24 bits)
O	LPDWORD	**NewValue**	New value of the counter (depend of the card and mode)
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : Increase

Example: (Decrease counter 1 in valid session (GTML))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;
DWORD               value=0x101010;
DWORD               newValue
BYTE                lnOut;
BYTE                bufOut[LN_OUT_MAX];

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID= GTML_SID_RT_EVENTS_LOG;
Secur.LID=0x00;
Secur.NKEY=RT_TAC_KEY;

ret=OpenSession(01,Secur,01,&Session,&Status);
Secur.SID=GTML_SID_RT_COUNTER_1;
ret=Decrease(Secur,0,value,&newValue,&Status);
ret=CloseSession(bufOut,&lnOut,&Status);
```

ReadRecord

Description : Reads record in circular or linear EF.

Syntax : DWORD ***ReadRecord*** (sCARD_SecurParam **SecurParam,**
BYTE **NuRec,**
BYTE **DataLen,**
LPBYTE **Data,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of data to be read
O	LPBYTE	**Data**	Read data
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: AppendRecord, UpdateRecord, SelectFile

Example: (ReadRecord stamped mode (CD97))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
BYTE                bufOut[0x1D];

Secur.AccMode=GEN_ACCESS_MODE_STAMPED;
Secur.SID=0x18;
Secur.LID=0x3120;
Secur.NKEY=MPP_LOG_KEY;
Secur.RFU=0x00;

ret=ReadRecord(Secur,1,0x1D,bufOut,&Status);
```

ReadRecordMultiple

Description : Reads record in circular or linear EF.

Syntax : DWORD ***ReadRecordMultiple*** (sCARD_SecurParam **SecurParam,**
BYTE **NuRec,**
BYTE **DataLen,**
BYTE **RecordOffset,**
BYTE **RecordLenght,**
LPBYTE **Data,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of data to be read
I	BYTE	**RecordOffset**	Offset in the records whare to start reading
I	BYTE	**RecordLenght**	Number of bytes to read from each record
O	LPBYTE	**Data**	Read data
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See also: ReadRecord, AppendRecord, UpdateRecord, SelectFile

AppendRecord

Description : Adds a record to circular EF.

Syntax : DWORD ***AppendRecord*** (sCARD_SecurParam **SecurParam,**
LPBYTE **Rec,**
BYTE **RecSize,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	LPBYTE	**Rec**	New record data
I	BYTE	**RecSize**	Length of new record data.
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : ReadRecord, UpdateRecord, SelectFile

Example: (AppendRecord in valid session (GTML))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;
BYTE                lnOut;
BYTE                bufOut[LN_OUT_MAX];
BYTE                bufIn[4]={0x11, 0x11, 0x11, 0x11};

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID= GTML_SID_RT_EVLOG;
Secur.LID= GTML_LID_RT_ EVLOG;
Secur.NKEY=RT_TAC_KEY;

ret=OpenSession(02,Secur,01,&Session,&Status);
ret=AppendRecord(Secur,bufIn,4,&Status);
ret=CloseSession(bufOut,&lnOut,&Status);
```

ChangeKey

Description : Change the key / Personnalization.

Syntax : DWORD ***ChangeKey*** (BYTE KeyIndex,
BYTE KeyIndexEncipher,
BYTE NewKeyVersion,
BYTE ALGTag,
BYTE ALGSam,
BYTE NewKeyIndex,
sCARD_Status* Status);

Parameters :

I	BYTE	**KeyIndex**	Index of the key (01 - 03)
I	BYTE	**KeyIndexEncipher**	Index of the key to encipher the transfer
I	BYTE	**NewKeyVersion**	New version of the key (<> 0)
I	BYTE	**ALGTag**	Algo key card to recopy
I	BYTE	**ALGSam**	Algo of the Sam used
I	BYTE	**NewKeyIndex**	index of the new key in the card in the DF
O	sCARD_Status	***Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

UpdateRecord

Description : Erases and writes a record in an EF.

Syntax : DWORD ***UpdateRecord*** (sCARD_SecurParam **SecurParam,**
BYTE **NuRec,**
BYTE **DataLen,**
LPBYTE **Data,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of data to write
I	LPBYTE	**Data**	Data to write
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : ReadRecord, AppendRecord, SelectFile, WriteRecord

Example: (UpdateRecord in reload session (GTML))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;
BYTE                lnOut;
BYTE                bufOut[LN_OUT_MAX];
BYTE                bufIn[4]={0x11, 0x11, 0x11, 0x11};

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID= GTML_SID_RT_COUNTER1;
Secur.LID= GTML_LID_RT_ COUNTER1;
Secur.NKEY=RT_RLD_KEY;

ret=OpenSession(01,Secur,01,&Session,&Status);
ret=UpdateRecord(Secur,1,4,bufIn,&Status);
ret=CloseSession(bufOut,&lnOut,&Status);
```

WriteRecord

Description : Writes a record in an EF.

Syntax : DWORD ***WriteRecord*** (sCARD_SecurParam **SecurParam,**
BYTE **NuRec,**
BYTE **DataLen,**
LPBYTE **Data,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**NuRec**	Record number
I	BYTE	**DataLen**	Length of data to write
I	LPBYTE	**Data**	Data to write
O	sCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : ReadRecord, AppendRecord, UpdateRecord

Example: (WriteRecord in valid session (GTML))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;
BYTE                lnOut;
BYTE                bufOut[LN_OUT_MAX];
BYTE                bufIn[4]={0x11, 0x11, 0x11, 0x11};

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID= GTML_SID_RT_EVLOG;
Secur.LID= GTML_LID_RT_ EVLOG;
Secur.NKEY=RT_TAC_KEY;

ret=OpenSession(02,Secur,01,&Session,&Status);
ret=WriteRecord(Secur,1,4,bufIn,&Status);
ret=CloseSession(bufOut,&lnOut,&Status);
```

OpenSession

Description : Opens a secured session in simple mode.

Syntax : DWORD ***OpenSession*** (BYTE **Type,**
sCARD_SecurParam **SecurParam,**
BYTE **Nrec,**
sCARD_Session* **Session,**
sCARD_Status* **Status**) **;**

Parameters :

I	BYTE	**Type**	Operation type: - Personalization (0x00) - Reload (0x01) - Validation (0x02) (Or others, depends of the card)
I	SCARD _SecurParam	**SecurParam**	Security parameters: SID, NKEY.
I	BYTE	**Nrec**	Record number
O	SCARD _Session*	**Session**	Application data return value
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : CloseSession

Example: (OpenSession in valid on counter1 (CD97))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID= GTML_SID_RT_COUNTER1;
Secur.LID= GTML_LID_RT_ COUNTER1;
Secur.NKEY=RT_TAC_KEY;

ret=OpenSession(02,Secur,01,&Session,&Status);
```

OpenSessionExt

Description : Opens a secured session in extended mode

Syntax : DWORD ***OpenSessionExt*** (BYTE **Type,**
sCARD_SecurParam **SecurParam,**
BYTE **Nrec,**
BYTE* **KVC,**
sCARD_Session* **Session,**
sCARD_Status* **Status**) **;**

Parameters :

I	BYTE	**Type**	Operation type: - Personalization (0x00) - Reload (0x01) - Validation (0x02) (Or others, depends of the card)
I	SCARD _SecurParam	**SecurParam**	Security parameters: SID, NKEY.
I	BYTE	**Nrec**	Record number
O	BYTE*	**KVC**	Card KVC returned
O	SCARD _Session*	**Session**	Application data return value
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : CloseSession

Example: (OpenSessionExt in valid on counter1 (GTML2))

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
sCARD_Session       Session;
BYTE                KVC;
Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID= GTML_SID_RT_COUNTER1;
Secur.LID= GTML_LID_RT_ COUNTER1;
Secur.NKEY=RT_TAC_KEY;
ret=OpenSessionExt(02,Secur,01,&KVC,&Session,&Status);
```

CloseSession

Description : Closes a secured session

Syntax : DWORD ***CloseSession*** (LPBYTE **Result,**
LPDWORD **cbResult,**
sCARD_Status* **Status**) **;**

Parameters :

o	LPBYTE	**Result**	Order result
o	LPDWORD	**cbResult**	Length of order result
o	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : OpenSession

Example:

```
DWORD           ret;
sCARD_Status    Status;
BYTE            lnOut;
BYTE            bufOut[LN_OUT_MAX];

ret=CloseSession(bufOut,&lnOut,&Status);
```

AbortSecuredSession

Description: Stop the current certification session. This still allow to continue a dialogue with the badge and, in particular, open a new session

Syntax: DWORD ***AbortSecuredSession*** (sCARD_Status* **Status**);

Parameters:

o	sCARD_Status	***Status**	Contains the card execution return status

Return: Return Value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Lock_Unlock

Description : Lock or Unlock the card

Syntax : DWORD ***Lock_Unlock*** **(** BYTE **Type,**
sCARD_Status* **Status**) **;**

Parameters :

I	BYTE	**Type**	00 = Lock the card. 01 = Unlock the card. Warning : this command success only if you had done a good presentation of the PIN before.
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example : (lock DF RT - CT2000)

```
DWORD               ret;
sCARD_SecurParam    Secur;
sCARD_Status        Status;
BYTE                PIN[4] = {0x30, 0x30, 0x30, 0x30};
BYTE                SEL_2000[]={0x20,0x00};
BYTE                bufOut[LN_OUT_MAX];

Secur.AccMode=0x00;
Secur.RFU=0x00;
Secur.SID=0x00;
Secur.LID=0x00;
Secur.NKEY=MF_INV_KEY;      // in crypted mode. (0x00 for clear mode).

ret=VerifyPIN(Secur, PIN, &Status);
ret=SelectFile(GEN_SEL_PATH,SEL_2000,sizeof(SEL_2000),bufOut,&Status);
ret=Lock_Unlock(00,&Status);
```

Multi_Decrease

Description : Decreases many counters on the same time.

Syntax : DWORD ***Multi_Decrease*** (sCARD_SecurParam **SecurParam,**
BYTE **NumberCpt,**
LPBYTE **Data,**
LPBYTE **NewData,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**NumberCpt**	Number of counters (7 Max).
I	LPBYTE	**Data**	Value to deduce (Lng Data = NumberCpt x 4) **Format of each 4 bytes bloc:** byte 1 : counter number. byte 2-4 : Value to deduce of the selected counter.
O	LPBYTE	**NewData**	New value of the counters: **Format of each 4 bytes bloc:** byte 1 : counter number. byte 2-4 : New value of the selected counter.
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : Multi_Increase

Multi_Increase

Description : Increases many counters on the same time.

Syntax : DWORD ***Multi_Increase*** (sCARD_SecurParam **SecurParam,**
BYTE **NumberCpt,**
LPBYTE **Data,**
LPBYTE **NewData,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID, LID, NKEY.
I	BYTE	**NumberCpt**	Number of counters (7 Max).
I	LPBYTE	**Data**	Value to add (Lng Data = NumberCpt x 4) **Format of each 4 bytes bloc:** Octet 1 : counter number. Octets 2-4 : Value to add of the selected counter.
O	LPBYTE	**NewData**	New value of the counters: **Format of each 4 bytes bloc:** Octet 1 : counter number. Octets 2-4 : New value of the selected counter.
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : Multi_Decrease

GetEPStatus_CD97

Description : Returns the EP value and prepares for purchasing or reloading.

Syntax : DWORD ***GetEPStatus_CD97*** (sCARD_SecurParam **SecurParam,**
BYTE **Type,**
LPDWORD **EP,**
LPBYTE **Log,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: NKEY (Key number to use in the SAM).
I	BYTE	**Type**	Transaction type to achieve 00 = Reload transaction 01 = Purchase transaction 02 = Cancel purchase
O	LPDWORD	**EP**	EP value
O	LPBYTE	**Log**	If Type = 00 alors Log = Reload log (22 car.) If Type = 01 alors Log = Paid log (19 car.) If Type = 02 alors Log = Paid log (19 car.)
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : Purchase_CD97

Purchase_CD97

Description : Purchases with EP.

Syntax : DWORD ***Purchase_CD97*** (BYTE **Type,**
LPBYTE **DataLog,**
LPBYTE **Disp,**
sCARD_Status* **Status**) **;**

Parameters :

I	BYTE	**Type**	Purchase Type 00 = purchase Transaction 01 = purchase Transaction with display
I	LPBYTE	**DataLog**	7 first bytes of payments' log new recording.
I	LPBYTE	**Disp**	6 bytes display
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : GetEPStatus_CD97

ReloadEP_CD97

Description : Reloads EP.

Syntax : DWORD ***ReloadEP_CD97*** (LPBYTE **ChargLog1,**
LPBYTE **ChargLog2,**
sCARD_Status* **Status**) **;**

Parameters :

I	LPBYTE	**ChargLog1**	5 first bytes of new reload log record : - Date (2 bytes) number of days since 1/1/97 - Money batch (2 octets) - Equipment type (1 octet)
I	LPBYTE	**ChargLog2**	5 bytes, offset [0x08..0x13] new reload long record : - Amount (3 octets) - Time (2 octets) number of minutes since 00 :00 :00
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : GetEPStatus_CD97

CancelPurchase_CD97

Description : Cancels the last payment made by EP.

Syntax : DWORD ***CancelPurchase_CD97 (*** BYTE **Type,**
LPBYTE **DataLog,**
LPBYTE **Disp,**
sCARD_Status* **Status**) **;**

Parameters :

I	BYTE	**Type**	Purchase Type 00 = purchase Transaction 01 = purchase Transaction with display
I	LPBYTE	**DataLog**	7 first bytes of payments' log new recording
I	LPBYTE	**Disp**	6 bytes display
O	SCARD _Status*	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

See Also : Purchase_CD97

DecreaseLG

Description : Decreases the value contained in a counter file and writes the 5 free data. Records the associated data. It is a command for CD97 card only.

Syntax : DWORD ***DecreasedLG*** (sCARD_SecurParam ***Secur,***
BYTE ***ICount,***
LPBYTE ***Value,***
sCARD_Status**** Status,***
LPDWORD ***NewValue***);

Parameters :

I	sCARD_Secur Param	***Secur***	Contain the parameters for the security - AccMode : Card Access Mode (GEN_ACCESS_MODE_DEFAULT, ...) - SID : Short ID Number (ex. : CD97_SID_RT_JOURNAL, ...) - LID : Long ID - NKEY: Number of Key which use in the SAM (in future KIF) - RUF : Reserved for KVC
I	BYTE	***ICount***	Index of the counter
I	LPBYTE	***Value***	Value to decreased(3 bytes, binary number positive or nil) + 5 free bytes
O	SCARD_Status	****Status***	Contains the card execution return status
O	LPDWORD	***NewValue***	Counter new value (Out of sessions Mode)(3 bytes, binary number signed)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

IncreaseLG

Description : Decreases the value contained in a counter file and writes the 5 free data. Records the associated data. It is a command for CD97 card only.

Syntax : DWORD ***IncreasedLG*** (sCARD_SecurParam ***Secur,***
BYTE ***ICount,***
LPBYTE ***Value,***
sCARD_Status**** Status,***
LPDWORD ***NewValue***);

Parameters :

I	sCARD_Secur Param	***Secur***	Contain the parameters for the security - AccMode : Card Access Mode (GEN_ACCESS_MODE_DEFAULT, ...) - SID : Short ID Number (ex. : CD97_SID_RT_JOURNAL, ...) - LID : Long ID - NKEY: Number of Key which use in the SAM (in future KIF) - RUF : Reserved for KVC
I	BYTE	***ICount***	Index of the counter
I	LPBYTE	***Value***	Value to decreased(3 bytes, binary number positive or nil) + 5 free bytes
O	SCARD_Status	****Status***	Contains the card execution return status
O	LPDWORD	***NewValue***	Counter new value (Out of sessions Mode)(3 bytes, binary number signed)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

ReadBinary

Description : Reading of a part of the content of a binary structure EF.

Syntax : DWORD ***ReadBinary*** (sCARD_SecurParam **SecurParam,**
BYTE **OffsetMSB,**
BYTE **OffsetLSB,**
BYTE **DataLen,**
LPBYTE **Data,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID
I	BYTE	**OffsetMSB**	MSB of the offset of the first byte (valid if SID = $00)
I	BYTE	**OffsetLSB**	LSB of the offset of the first byte
I	BYTE	**DataLen**	length of data to read
O	LPBYTE	**Data**	Read data
O	sCARD _Status*	**Status**	Execution status (depend of the card)

UpdateBinary

Description : Deletion then writing of a a part of the content of a binary structure EF.

Syntax : DWORD ***UpdateBinary*** (sCARD_SecurParam **SecurParam,**
BYTE **OffsetMSB,**
BYTE **OffsetLSB,**
BYTE **DataLen,**
LPBYTE **Data,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID
I	BYTE	**OffsetMSB**	MSB of the offset of the first byte (valid if SID = $00)
I	BYTE	**OffsetLSB**	LSB of the offset of the first byte
I	BYTE	**DataLen**	length of data to read
I	LPBYTE	**Data**	Data to write
O	sCARD _Status*	**Status**	Execution status (depend of the card)

WriteBinary

Description : This command is used to set to 1 the bits in a part of the content of a binary structure EF. This can be considered like "OR-ing" the bits already present with the data bits in the command

Syntax : DWORD ***WriteBinary*** (sCARD_SecurParam **SecurParam,**
BYTE **OffsetMSB,**
BYTE **OffsetLSB,**
BYTE **DataLen,**
LPBYTE **Data,**
sCARD_Status* **Status**) **;**

Parameters :

I	SCARD _SecurParam	**SecurParam**	Security parameters: AccMode, SID
I	BYTE	**OffsetMSB**	MSB of the offset of the first byte (valid if SID = $00)
I	BYTE	**OffsetLSB**	LSB of the offset of the first byte
I	BYTE	**DataLen**	length of data to read
I	LPBYTE	**Data**	Data to write
O	sCARD _Status*	**Status**	Execution status (depend of the card)

CalypsoRev3_SelectApplication

Description : Select an Application for Calypso rev3

N.B.: This command affects the APDU Class permanently (see command 01_0C to set it back to proprietary class)

Syntax : DWORD ***WriteBinary*** (BYTE **FCIChoice,**
BYTE **LgAID,**
LPBYTE **AID,**
sCARD_Status* **Status**
LPBYTE **FCI)**

Parameters :

I	BYTE	**FCIChoice**	Choice for the FCI returned : 0 = current AID 2= No FCI returned
I	BYTE	**LgAID**	Length of the following AID
I	LPBYTE	**AID**	Application ID (variable length)
O	sCARD_Status *	**Status**	Execution status (depend of the card)
I	LPBYTE	**Data**	Depend on the FCI choice input flag

CheckCertificate

Description : Controls the certificate.

Syntax : DWORD ***CheckCertificate*** (BYTE ***KeyType***,
BYTE ***Param***,
BYTE ***LngBuffer***,
LPBYTE ***Buffer***,
BYTE ***LngCertificat***,
BYTE *****Certificat***,
BYTE *****Status***);

Parameters :

I	BYTE	***KeyType***	Number or type of the key
I	BYTE	***Param***	Algorithm type
I	BYTE	***LngBuffer***	Length of the buffer (8 bytes of diversifier + n bytes of data)
I	LPBYTE	***Buffer***	Data to control
I	BYTE	***LngCertificat***	Length of the certificate (2 or 4 bytes)
I	BYTE *	***Certificat***	Certificate read in the CTx
O	BYTE *	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        RFU=0x00;
BYTE        dataToControl[LN_IN]=DATA_TO_CONTROL;
BYTE        certificateInTick[LN_CERTIF]=CERTIFICATE;

ret= CheckCertificate (TIC_PUR_KEY,RFU, LN_IN, dataToControl, LN_CERTIF, certificateInTick, &status );
```

GiveCertificate

Description : Generates the certificate.

Syntax : DWORD ***GiveCertificate*** (BYTE ***KeyType***,
BYTE ***Param***,
BYTE ***LngBuffer***,
LPBYTE ***Buffer***,
BYTE ***LngCertificat***,
BYTE *****Certificat***,
BYTE *****Status***);

Parameters :

I	BYTE	***KeyType***	Number or type of the key
I	BYTE	***Param***	Algorithm type
I	BYTE	***LngBuffer***	Length of the buffer (8 bytes of diversifier + n bytes of data)
I	LPBYTE	***Buffer***	Data to control
I	BYTE	***LngCertificat***	Length of the certificate (2 or 4 bytes)
O	BYTE *	***Certificat***	Calculated certificate
O	BYTE *	**Status**	Execution status (depend of the card)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        RFU=0x00;
BYTE        dataToControl[LN_IN]=DATA_TO_CONTROL;
BYTE        certificateOut[LN_CERTIF];

ret= GiveCertificate (TIC_RLD_KEY,RFU, LN_IN, dataToControl, LN_CERTIF, certificateOut, &status );
```

1.12. CTx card management functions

1.12.1. CTS256B functions

CTx_Active

Description : Actives CTx ticket and responds 5 first blocs (equivalent to EnterHuntPhase).

Syntax : DWORD **CTx_Active** (LPBYTE ***Data***,
BYTE* ***Status***)

Parameters :

o	LPBYTE	**Data**	Data read
o	BYTE *	**Status**	Status communication to the instruction $00: interrupted communication $01: bad CRC $0F: success $40: a card is detected but it is not a CTx $80: collision

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataReadActive[LN_OUT_ACTIVE];

ret= CTx_Active(dataReadActive, &status );
```

CTx_Read

Description : Reads at given address a given number of bytes.

Syntax : DWORD **CTx_Read** (BYTE ***ADD***,
BYTE ***NB***,
LPBYTE ***Data***,
BYTE* ***Status***)

Parameters :

I	BYTE	***ADD***	Address of the first read (0 …31) in byte
I	BYTE	***NB***	Number of byte to be read from 1 to 32
O	LPBYTE	**Data**	Data read
O	BYTE *	**Status**	Status communication to the instruction $00: interrupted communication $01: bad CRC $02: success

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataRead[10];
BYTE        dataReadActive[LN_OUT_ACTIVE];

ret= CTx_Active(dataReadActive, &status );
ret= CTx_Read(02,10,dataRead, &status );
```

CTx_Update

Description : Erases if needed, writes then verifies at given address a given number of byte.

Syntax : DWORD **CTx_Update** (BYTE ***ADD***,
BYTE ***NB***,
LPBYTE ***DataToWrite***,
LPBYTE ***DataInCTS,***
LPBYTE ***Data,***
BYTE* ***Status***)

Parameters :

I	BYTE	***ADD***	Address of the first byte to write (0 …31)
I	BYTE	***NB***	Number of byte to write from 1 up to 32
I	LPBYTE	***DataToWrite***	Data to be written
I	LPBYTE	***DataInCTS***	Data read (or 0xEE if unknow) in the CTx
O	LPBYTE	**Data**	Data read
O	BYTE *	**Status**	Status communication to the instruction $00: interrupted communication $01: bad CRC $02: success

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataRead[4];
BYTE        dataToWrite[4]={0x11, 0x11, 0x11, 0x11};
BYTE        dataInTick[4]={0xEE, 0xEE, 0xEE, 0xEE};
BYTE        dataReadActive[LN_OUT_ACTIVE];
BYTE        dataSyst[SYSTBITS_LENGTH];

ret= CTx_Active(dataReadActive, &status );
ret= CTx_Read(ADD_SYSTBITS, SYSTBITS_LENGTH, dataSyst, &status);      // activation for updating
ret= CTx_Update(10,4,dataToWrite,dataInTick,dataRead,&status );
```

CTx_Release

Description : Desactivates CTx.

Syntax : DWORD **CTx_*Release*** (BYTE ***Param***,
BYTE* ***Status***)

Parameters :

I	BYTE	***Param***	0x00 (desactivate ticket with instruction desactivate) others = RFU
O	BYTE *	**Status**	Status communication to the instruction 00 : still active ticket 02 : no more active ticket

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
BYTE        status;
DWORD       ret;

ret=CTx_Release(0x00,&status);
```

1.12.2. CTx512x functions

CTx512x_List (CTx512B only)

Description : Performs anticollision and answers the serial numbers of all the chips present in the reader field.

After *LIST* instruction, all the tickets are in *HALT* state. Each ticket has to be selected by its serial number before any other command.

Syntax :

DWORD ***CTx512x_List*** (BYTE ***RFU***,
BYTE* ***nbTickets,***
BYTE* ***serialNumbers***,
BYTE* ***status***);

Parameters :

I	BYTE	***RFU***	0x00 (others RFU)
O	BYTE*	***nbTickets***	number of tickets in the antenna field
O	BYTE*	***serialNumbers***	list of the serial numbers retrieved (2 LSB serial number bytes for each ticket : address $03)
O	BYTE *	***status***	CTx512B execution status returned : $00 : no ticket $x1 : CTx512B in antenna field $x2 : CTS256B in antenna field $x3 : both CTx512B and CTS256B in antenna field $x4 : identification error (chip version or manufacturer) $x5 : mysterious answer $8x : timeout reached before end of anticollision : problem occurs

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        nbTickets;
BYTE        serialNumbers[2*NB_TICK_MAX];

ret=CTx512x_List(RFU,&nbTickets,serialNumbers,&status);
```

CTx512x_Select (CTx512B only)

Description : Selects a specific ticket by its serial number.

Syntax : DWORD ***CTx512x_Select*** (BYTE* ***serialNumber,***
BYTE* ***serialNumberRead,***
BYTE* ***status***);

Parameters :

I	BYTE*	***serialNumber***	pointer to the buffer containing the serial number (2 bytes)
O	BYTE*	***serialNumberRead***	pointer to serial number read (2 bytes) (should be equal to serialNumber)
O	BYTE *	***status***	CTX512B execution status returned : $00 : no answer $01 : bad CRC $02 : success

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        nbTickets;
BYTE        serialNumber[2];
BYTE        serialNumberRead[2];
BYTE        serialNumbers[2*NB_TICK_MAX];

ret=CTx512x_List(RFU,&nbTickets,serialNumbers,&status);
serialNumber[0]=serialNumbers[0];       // copy first serial number
serialNumber[1]=serialNumbers[1];       // for selection
ret=CTx512x_Select(serialNumber,serialNumberRead,&status);
```

CTx512x_Read

Description : Reading of a number of bytes at a given address.

Internally, the reader chooses read or multiread instruction depending on NB parameter.

Syntax : DWORD ***CTx512x_Read*** (BYTE ***ADD***,
BYTE ***NB***,
BYTE**** dataRead,***
BYTE* ***status***);

Parameters :

I	BYTE	***ADD***	address of the first byte (0 ... 63)
I	BYTE	***NB***	number of bytes to be read (from 1 up to 64)
O	BYTE *	***dataRead***	pointer to data read buffer
O	BYTE *	***status***	CTx512x execution status returned : $00 : no answer $01 : bad CRC $02 : success $03 : bad parameters

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataRead[4];

ret= CTx512x_Read(10,4,dataRead, &status );
```

CTx512x_Update

Description : Deletes, writes, then checks by reading the written bytes.

Syntax : DWORD ***CTx512x_Update***(BYTE ***ADD,***
BYTE ***NB***,
BYTE* ***dataToUpdate***,
BYTE* ***dataRead,***
BYTE* ***status***);

Parameters :

I	BYTE	***ADD***	address of the first byte to update (0 ... 63)
I	BYTE	***NB***	number of bytes to be updated (from 1 up to 64)
I	BYTE *	***DataToUpdate***	Pointer to the *dataToUpdate* buffer
O	BYTE *	***dataRead***	pointer to the dataRead buffer
O	BYTE *	***status***	CTX512X execution status returned : $00 : no answer $01 : bad CRC $02 : success $03 : bad parameters $83 : security activated (ie: data written =/= dataRead)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataRead[4];
BYTE        dataToUpdate [4]={0x11, 0x11, 0x11, 0x11};

ret= CTx512x_Update(10,4,dataToUpdate, dataRead,&status );
```

CTx512x_Write

Description : Writes (logical 'OR'), then checks by reading the written bytes. Useful for the OTP zone.

Syntax :

DWORD ***CTx512x_Write*** (BYTE ***ADD,***
BYTE ***NB***,
BYTE* ***dataToWrite***,
BYTE* ***dataRead,***
BYTE* ***status***);

Parameters :

I	BYTE	***ADD***	address of the first byte to write (0 ... 63)
I	BYTE	***NB***	number of bytes to be written (from 1 up to 64)
I	BYTE *	***DataToWrite***	Pointer to the *dataToWrite* buffer
O	BYTE *	***dataRead***	pointer to the dataRead buffer
O	BYTE *	***status***	CTX512B execution status returned : $00 : no answer $01 : bad CRC $02 : success $03 : Bad parameters $82 : security activated (ie: data written =/= dataRead)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataRead[4];
BYTE        dataToWrite [4]={0x11, 0x11, 0x11, 0x11};

ret= CTx512x_Write(10,4,dataToWrite, dataRead,&status );
```

CTx512x_Halt

Description : Halt CTx512x ticket.

Syntax : DWORD ***CTx512x_Halt*** (BYTE ***param,***
BYTE* ***status***);

Parameters :

I	BYTE	***param***	$00 : desactivates using the “desactivate” instruction. (others RFU)
O	BYTE *	***status***	CTX512X execution status returned : $00 : ticket still active $02 : ticket desactivated

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
BYTE        status;
DWORD       ret;

ret=CTx512x_Halt(0x00,&status);
```

CTx512x_Authenticate (CTM512B only)

Description : Authenticates an area of the CTM512B (8 consecutive bytes).

Syntax : DWORD ***CTx512x_Authenticate***(BYTE ***ADD***,
BYTE ***kif_kref,***
BYTE ***kvc_zero***,
BYTE* ***status,***
BYTE* ***dataSAMLength***,
BYTE* ***dataSAM***);

Parameters :

I	BYTE	***ADD***	Starting address of the area to authenticate
I	BYTE	***kif_kref***	specifies the KIF or the key reference (if key reference used, kvc_zero must be set to 0x00)
I	BYTE	***kvc_zero***	specifies the KVC if the KIF has been specified in kif_kref (if KIF has not been specified in kif_kref, must be set to 0x00)
O	BYTE *	***status***	CTX512B execution status returned : $00 : no answer $01 : unexpected failure $02 : success $03 : Bad parameters $04 : no current SAM $05 : SAM not initialized $06 : bad SAM status
O	BYTE*	***dataSAMLength***	Length of the data returned by the SAM.
O	BYTE*	***dataSAM***	Buffer with the data returned by the SAM (only when status = $06, ie an error occured)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataSAMLength;
BYTE        dataSAM[MAX_ANSWER_LENGTH];

ret = CTx512x_Authenticate(0x05, 0x30, 0x00, &status, &dataSAMLength, dataSAM);
```

CTx512x_WriteKey (CTM512B only)

Description : Compute and write the key in the CTM512B. The algorithm (perso fuse) is then activated.

Syntax : DWORD ***CTx512x_WriteKey*** (BYTE ***kif_kref,***
BYTE ***kvc_zero***,
BYTE* ***status,***
BYTE* ***dataSAMLength***,
BYTE* ***dataSAM***);

Parameters :

I	BYTE	***kif_kref***	specifies the KIF or the key reference (if key reference used, kvc_zero must be set to 0x00)
I	BYTE	***kvc_zero***	specifies the KVC if the KIF has been specified in kif_kref (if KIF has not been specified in kif_kref, must be set to 0x00)
O	BYTE *	***status***	CTX512B execution status returned : $00 : no answer $01 : unexpected failure $02 : success $03 : Bad parameters $04 : no current SAM $05 : SAM not initialized $06 : bad SAM status $07 : writing prohibited
O	BYTE*	***dataSAMLength***	Length of the data returned by the SAM.
O	BYTE*	***dataSAM***	Buffer with the data returned by the SAM (only when status = $06, ie an error occured)

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The receive data is wrong
RCSC_CheckSum	CRC Error

Example:

```
DWORD       ret;
BYTE        status;
BYTE        dataSAMLength;
BYTE        dataSAM[MAX_ANSWER_LENGTH];

ret = CTx512x_WriteKey(0x30, 0x00, &status, &dataSAMLength, dataSAM);
```

1.13. MIFARE® card functions

Functions in this section allow managing all necessary operations on a MIFARE® Classic card assuming that the specific hardware module is present in the CSC.

For all of these functions the error code table is listed here-under :

This function class is limited to the command available on the MIFARE® Classic card.

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The received data are wrong
RCSC_CheckSum	CRC Error

MIFARE_LoadReaderKeyIndex

Description : Load a MIFARE® Key in the reader: The reader contains 32 locations to store keys in the secure module EEPROM. This secure module can then refer to theses keys by an index to perform cryptographic calculations (the indexed key is then copied in RAM (keys are used by their index in the following DLL commands: **MIFARE_ChangeKey**, **MIFARE_Authenticate** & **MIFARE_ReadSector** which perform a sector authentication and allow the use of other DLL commands ReadBlock, WriteBlock, IncrementValue, DecrementeValue & BackupRestoreValue).

Syntax : DWORD ***MIFARE_LoadReaderKeyIndex*** (BYTE ***KeyIndex***,
LPBYTE ***KeyVal***,
LPBYTE ***Status***)

Parameters :

I	BYTE	***KeyIndex***	Index of the key in the reader EEPROM 0 to 31
I	LPBYTE	***KeyVal***	Key Value (6 octets)
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD       ret;
BYTE        keyVal[6]={0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
BYTE        Status;
ret=MIFARE_LoadReaderKeyIndex(0, keyVal, &Status);
```

MIFARE_ChangeKey

Description : Change of key in the MIFARE card sector.

Syntax : DWORD ***MIFARE_ChangeKey*** (BYTE ***InitialKeyAorB***,
BYTE ***NumSector***,
BYTE ***InitialKeyIndex***,
BYTE ***FinalKeyAorB,***
LPBYTE ***NewKeyA***,
LPBYTE ***NewAccessBits***,
LPBYTE ***NewKeyB***,
LPBYTE ***MifareType***,
LPBYTE ***SerialNumber***,
LPBYTE ***Status***)

Parameters :

I	BYTE	***InitialKeyAorB***	Type of key A or B used to authenticate before the change command
I	BYTE	***NumSector***	sector N° from 0 to 15
I	BYTE	***InitialKeyIndex***	Key Index in the reader 0 to 31 to authenticate before change command
I	BYTE	***FinalKeyAorB***	Type of key A or B to be used after change command
I	LPBYTE	***NewKeyA***	New value for A Key (6 bytes) in reverse byte order
I	LPBYTE	***NewAccessBits***	New value for the 4 Access Bits Bytes (3+1 free)
I	LPBYTE	***NewKeyB***	New value for B Key (6 bytes) in reverse byte order
O	LPBYTE	***MifareType***	Answers the Card type (08 for MIFARE® Classic)
O	LPBYTE	***SerialNumber***	Answers the serial number of the card (4 bytes)
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD       ret;
BYTE        NewKeyA[6]={0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
BYTE        NewKeyB[6]={0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
BYTE        NewAccessBits[6]={0xFF,0x07,0x80,0x00};
BYTE        MifareType;
BYTE        SerialNb[4];
BYTE        Status;
ret=MIFARE_ChangeKey(0x0A,1,0,0x0B,NewKeyA,NewAccessBits,NewKeyB,&MifareType,SerialNb,&Status);
```

MIFARE_Select

Description : Selects a MIFARE card with its unique ID. Enables to detect a card in case of collision. This command requires the MF RC500 chip. It realizes the ISO 14443 connexion such as REQA and SELECT.

Syntax : DWORD ***MIFARE_Select*** (BYTE* ***SerialNumber***,
BYTE ***SerialNumberLn***,
BYTE* ***Status,***
BYTE****** ***SerialNumberOut***)

Parameters :

I	BYTE*	***SerialNumber***	Buffer containing the serial Number of the card to select
I	BYTE	***SerialNumberLn***	Length of the serial number
O	BYTE*	***Status***	Execution Report See Status values in the interface CSC document
O	BYTE*	***SerialNumberOut***	Buffer containing the serial Number of the card selected

Example:

```
DWORD               ret;
BYTE                MifareType;
BYTE                SerialNb[4];
BYTE                SerialNbRead[4];
BYTE                Status;
sCARD_SearchExt     SearchExt;
DWORD               search_mask;
BYTE                COM;
BYTE                atr[ATR_LENGTH_MAX];
DWORD               atrLength;

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCardExt(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
```

MIFARE_Authenticate

Description : Authenticate a given sector.

Syntax : DWORD ***MIFARE_Authenticate*** (BYTE ***NumSector***,
BYTE ***KeyAorB***,
BYTE ***KeyIndex***,
LPBYTE ***MifareType***,
LPBYTE ***SerialNumber***,
LPBYTE ***Status***)

Parameters :

I	BYTE	***NumSector***	Sector number from 1 to 15
I	BYTE	***KeyAorB***	Key type A or B
I	BYTE	***KeyIndex***	Index of the key in the reader from 0 to 31
O	LPBYTE	***MifareType***	Answers the card type (08 for MIFARE® Classic)
O	LPBYTE	***SerialNumber***	Answers the card serial number (4 bytes)
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD               ret;
BYTE                MifareType;
BYTE                SerialNb[4];
BYTE                SerialNbRead[4];
BYTE                Status;
sCARD_SearchExt     SearchExt;
DWORD               search_mask;
BYTE                COM;
BYTE                atr[ATR_LENGTH_MAX];
DWORD               atrLength;

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCard(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
ret=MIFARE_Authenticate(0,0x0A,0,&MifareType,SerialNbRead,&Status);
```

MIFARE_Halt

Description : De-activation of a MIFARE® card.

Syntax : DWORD ***MIFARE_Halt*** (void)

Parameters : void

Example:

```
DWORD                ret;
ret=MIFARE_Halt();
```

MIFARE_ReadBlock

Description : Read data in a given 16 bytes block (0 to 63) (Length is fixed).

Syntax : DWORD ***MIFARE_ReadBlock*** (BYTE ***NumBlock***,
LPBYTE ***DataRead,***
LPBYTE ***Status***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter from 0 to 63
O	LPBYTE	***DataRead***	Data read in the block (16 bytes)
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD                ret;
BYTE                 MifareType;
BYTE                 SerialNb[4];
BYTE                 SerialNbRead[4];
BYTE                 DataRead[16];
BYTE                 Status;
sCARD_SearchExt      SearchExt;
DWORD                search_mask;
BYTE                 COM;
BYTE                 atr[ATR_LENGTH_MAX];
DWORD                atrLength;
SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCard(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
ret=MIFARE_Authenticate(0,0x0A,0,&MifareType, SerialNbRead,&Status);
ret=MIFARE_ReadBlock(1,DataRead,&Status);
```

MIFARE_ReadSector

Description : Read a 64 bytes data area in a given Sector (0 to 15) (fixed length) with authentication of the sector.

Syntax : DWORD ***MIFARE_ReadSector*** (BYTE ***NumSector***,
BYTE ***KeyAorB***,
BYTE ***KeyIndex***,
LPBYTE ***MifareType,***
LPBYTE ***SerialNumber,***
LPBYTE ***DataRead,***
LPBYTE ***Status***)

Parameters :

I	BYTE	***NumSector***	Sector number from 0 to 15
I	BYTE	***KeyAorB***	Key type A or B
I	BYTE	***KeyIndex***	Index of the key in the reader from 0 to 31
O	LPBYTE	***MifareType***	Answers the card type (08 for MIFARE® Classic)
O	LPBYTE	***SerialNumber***	Answers the card serial number (4 bytes)
O	LPBYTE	***DataRead***	Data read in the whole sector (64 bytes)
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD           ret;
BYTE            MifareType;
BYTE            SerialNb[4];
BYTE            SerialNbRead[4];
BYTE            DataRead[64];
BYTE            Status;
sCARD_SearchExt SearchExt;
DWORD           search_mask;
BYTE            COM;
BYTE            atr[ATR_LENGTH_MAX];
DWORD           atrLength;

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCard(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
```

```
ret=MIFARE_ReadSector(0,0x0A,0,&MifareType, SerialNbRead,DataRead,&Status);
```

MIFARE_WriteBlock

Description : Write data in a given 16 bytes block (1 to 63) (Length is fixed) this command contains a re-read after write.

Syntax : DWORD ***MIFARE_WriteBlock*** (BYTE ***NumBlock***,
LPBYTE ***DataToWrite***,
LPBYTE ***DataVerif,***
LPBYTE ***Status***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter from 0 to 63
I	LPBYTE	***DataToWrite***	Data to write (16 bytes)
O	LPBYTE	***Data*Verif**	Data read in the block (16 bytes)
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD               ret;
BYTE                MifareType;
BYTE                SerialNb[4];
BYTE                SerialNbRead[4];
BYTE                DataRead[16];
BYTE DataToWrite[16]={0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01};
BYTE                Status;
sCARD_SearchExt     SearchExt;
DWORD               search_mask;
BYTE                COM;
BYTE                atr[ATR_LENGTH_MAX];
DWORD               atrLength;

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCard(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
ret=MIFARE_Authenticate(0,0x0A,0,&MifareType, SerialNbRead,&Status);
ret=MIFARE_WriteBlock(1,DataToWrite,DataRead,&Status);
```

MIFARE_DecrementValue

Description : Decrease a counter value.

Syntax : DWORD ***MIFARE_DecrementValue*** (BYTE ***NumBlock***, LPBYTE ***Substract***, LPBYTE ***Verif,*** LPBYTE ***Status***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter value from 1 to 63
I	LPBYTE	***Substract***	Value to decrease to the counter from 1 to 4294967295
O	LPBYTE	**Verif**	New counter value
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD               ret;
BYTE                MifareType;
BYTE                SerialNb[4];
BYTE                SerialNbRead[4];
BYTE                Status;
sCARD_SearchExt     SearchExt;
DWORD               search_mask;
BYTE                COM;
BYTE                atr[ATR_LENGTH_MAX];
DWORD               atrLength;
BYTE                decrement5[4] = {0x00,0x00,0x00,0x05};
BYTE                verif[5];

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCard(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
ret=MIFARE_Authenticate(0,0x0A,0,&MifareType, SerialNbRead,&Status);
ret=MIFARE_DecrementValue(2, decrement5,verif,&Status);
```

MIFARE_IncrementValue

Description : Increase a counter value.

Syntax : DWORD ***MIFARE_IncrementValue*** (BYTE ***NumBlock***, LPBYTE ***Addition***, LPBYTE ***Verif,*** LPBYTE ***Status***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter value from 1 to 63
I	LPBYTE	***Addition***	Value to add to the counter from 1 to 4294967295
O	LPBYTE	**Verif**	New counter value
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD               ret;
BYTE                MifareType;
BYTE                SerialNb[4];
BYTE                SerialNbRead[4];
BYTE                Status;
sCARD_SearchExt     SearchExt;
DWORD               search_mask;
BYTE                COM;
BYTE                atr[ATR_LENGTH_MAX];
DWORD               atrLength;
BYTE                increment3[4] = {0x00,0x00,0x00,0x03};
BYTE                verif[5];

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCard(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
ret=MIFARE_Authenticate(0,0x0A,0,&MifareType, SerialNbRead,&Status);
ret=MIFARE_IncrementValue(2, increment3,verif,&Status);
```

MIFARE_BackUpRestoreValue

Description : Copy the counter value in a backup location (the destination block must be in the same sector and configured also as a counter value).

Syntax : DWORD ***MIFARE_BackupRestoreValue*** (BYTE ***Origine***, BYTE ***Destination,*** LPBYTE ***Status***)

Parameters :

I	BYTE	***Origine***	Block number to copy
I	BYTE	***Destination***	Backup destination Block
O	LPBYTE	***Status***	Execution Report See Status values in the interface CSC document

Example:

```
DWORD               ret;
BYTE                MifareType;
BYTE                SerialNb[4];
BYTE                SerialNbRead[4];
BYTE                Status;
sCARD_SearchExt     SearchExt;
DWORD               search_mask;
BYTE                COM;
BYTE                atr[ATR_LENGTH_MAX];
DWORD               atrLength;

SearchExt.CONT=0x00;
SearchExt.ISOB=0x02;
SearchExt.ISOA=0x00;
SearchExt.TICK=0x00;
SearchExt.INNO=0x00;
SearchExt.MIFARE=0x01;
SearchExt.MV4k=0x00;
SearchExt.MV5k=0x00;
search_mask = SEARCH_MASK_MIFARE;

ret=CSC_SearchCard(&SearchExt,search_mask,0x01,0x44,&COM,&atrLength,atr);
memcpy(SerialNb,atr+2,4);       // copy serial number retreived by CSC_SearchCard
ret=MIFARE_Select(SerialNb,4, &Status, SerialNbRead);
ret=MIFARE_Authenticate(0,0x0A,0,&MifareType, SerialNbRead,&Status);
ret=MIFARE_BackUpRestoreValue(1, 2, &Status);
```

MIFARE_ReadMultipleBlock

Description : Read several blocks in an authenticated sector.

Syntax : DWORD ***MIFARE_ReadMultipleBlock*** (BYTE ***BlockNum***,
BYTE ***NumBlock***,
BYTE ****Status***,
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***BlockNum***	Block number from 0 to 255 (1 byte)
I	BYTE	***NumBlock***	Number of Block "n" (1 byte)
O	BYTE	****Status***	Execution Report See Status values in the interface CSC document
O	LPBYTE	***DataRead***	DataRead in the card (n x 16 bytes)

MIFARE_SimpleWriteBlock

Description : Write an authenticated block.

Syntax : DWORD ***MIFARE_SimpleWriteBlock*** (BYTE ***BlockNum***,
LPBYTE ***DataToWrite***,
BYTE ****Status***)

Parameters :

I	BYTE	***BlockNum***	Block number from 0 to 255 (1 byte)
I	LPBYTE	***DataToWrite***	Data to Write in the selected authenticated block (16 bytes)
O	BYTE	****Status***	Execution Report See Status values in the interface CSC document

MIFARE_ReadSectorData

Description : Read a the data blocks Sector of the PICC.

Syntax : DWORD ***MIFARE_ReadSectorData*** (BYTE ***KeyAorB***,
BYTE ***NumSector***,
BYTE ***KeyIndex***,
BYTE ****Status***,
BYTE ****MifareType***,
LPBYTE ***SerialNumber***,
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***KeyAorB***	Choice of the key needed for authentication (1 byte)
I	BYTE	***NumSector***	Sector to authenticate and read (1 byte)
I	BYTE	***KeyIndex***	Index from 0 to 31 of the Reader key used for authentication (1 byte)
O	BYTE	****Status***	Execution Report See Status values in the interface CSC document
O	BYTE	****MifareType***	Type of the card authenticated (08 for Mifare Classic) (1 byte)
O	LPBYTE	***SerialNumber***	Serial Number of the card authenticated (4 Bytes)
O	LPBYTE	***DataRead***	data read in the sector specified (48 bytes)

MIFARE_WriteSectorData

Description : Write a the data blocks Sector of the PICC.

Syntax : DWORD ***MIFARE_WriteSectorData*** (BYTE ***KeyAorB***,
BYTE ***NumSector***,
BYTE ***KeyIndex***,
LPBYTE ***DataToWrite***,
BYTE ***CardType***,
BYTE ****Status***)

Parameters :

I	BYTE	***KeyAorB***	Choice of the key needed for authentication (1 byte)
I	BYTE	***NumSector***	Sector to authenticate and read (1 byte)
I	BYTE	***KeyIndex***	Index from 0 to 31 of the Reader key used for authentication (1 byte)
I	LPBYTE	***DataToWrite***	data to write in the sector specified (48 bytes)

I	BYTE	***CardType***	Type of the card authenticated (08 for Mifare Classic) (1 byte)
O	BYTE	****Status***	Execution Report See Status values in the interface CSC document

1.13. MIFARE® - SAM NXP card functions

Functions in this section allow managing all necessary operations on a MIFARE® Classic card with external cryptography engine using NXP SAM instead of integrated RF Chip due to security improvement.

For all of these functions the error code table is listed here-under :
This function class is limited to the command available on the MIFARE® Classic card.

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The received data are wrong
RCSC_CheckSum	CRC Error

MIFARE_SAMNXP_Authenticate

Description : Authenticate a given block.

Syntax : DWORD ***MIFARE_SAMNXP_Authenticate*** (BYTE ***NumKey***,
BYTE ***VersionKey***,
BYTE ***KeyAorB***,
BYTE ***NumBlock***,
BYTE ***LgDiversifier***,
BYTE ***BlockDiversifier***,
BYTE ***LgDiversifier***,
BYTE ***BlockDiversifier,***
BYTE ***, ***StatusCard***,
WORD ***, ***StatusSam)***

Parameters :

I	BYTE	***NumKey***	Key Number (1 bytes)
I	BYTE	***VersionKey***	Key Version (1 bytes)
I	BYTE	***KeyAorB***	Key type A or B (1 bytes)
I	BYTE	***NumBlock***	Block number from 0 to 63 (1 bytes)
I	BYTE	***LgDiversifier***	Length Diversifier (1 bytes)
I	BYTE	***BlockDiversifier***	Block Diversifier (1 bytes)

O	BYTE	*__StatusCard__,	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	*__StatusSam__	Status Sam (2 bytes) : 0x9000 ➔ OK

MIFARE_SAMNXP_Re-Authenticate

Description : Re-Authenticate a given block.

Syntax : DWORD ***MIFARE_SAMNXP_ReAuthenticate*** (BYTE ***NumKey***,
BYTE ***VersionKey***,
BYTE ***KeyAorB***,
BYTE ***NumBlock***,
BYTE ***LgDiversifier***,
BYTE ***BlockDiversifier***,
BYTE ***LgDiversifier***,
BYTE ***BlockDiversifier,***
BYTE *__StatusCard__,
WORD *__StatusSam)__

Parameters :

I	BYTE	***NumKey***	Key Number (1 bytes)
I	BYTE	***VersionKey***	Key Version (1 bytes)
I	BYTE	***KeyAorB***	Key type A or B (1 bytes)
I	BYTE	***NumBlock***	Block number from 0 to 63 (1 bytes)
I	BYTE	***LgDiversifier***	Length Diversifier (1 bytes)
I	BYTE	***BlockDiversifier***	Block Diversifier (1 bytes)
O	BYTE	*__StatusCard__,	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	*__StatusSam__	Status Sam (2 bytes) : 0x9000 ➔ OK

MIFARE_SAMNXP_ReadBlock

Description : Read data in a given 16 bytes block (0 to 63) (Length is fixed).

Syntax : DWORD ***MIFARE_SAMNXP_ReadBlock*** (BYTE ***NumBlock***,
BYTE ****StatusCard,***
WORD ****StatusSam,***
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter from 0 to 63 (1 bytes)
O	BYTE	****StatusCard***	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK
O	BYTE	***DataRead***	Data read in the block (16 bytes)

MIFARE_ SAMNXP_WriteBlock

Description : Write data in a given 16 bytes block (1 to 63) (Length is fixed) this command contains a re-read after write.

Syntax : DWORD ***MIFARE_SAMNXP _WriteBlock*** (BYTE ***NumBlock***,
LPBYTE ***DataToWrite***,
BYTE ****StatusCard,***
WORD ****StatusSam,***
BYTE ****StatusWrite***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter from 0 to 63 (1 bytes)
I	LPBYTE	***DataToWrite***	Data to write (16 bytes)
O	BYTE	****StatusCard***	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK
O	BYTE	****StatusWrite***	Status Write (1 bytes) : 0x0A ➔ OK

MIFARE_ SAMNXP _ChangeKey

Description : Change of key in the MIFARE card block.

Syntax : DWORD ***MIFARE_SAMNXP _ChangeKey*** (BYTE ***NumKey***,
BYTE ***VersionKeyA***,
BYTE ***VersionKeyB***,
LPBYTE ***DefaultAccess,***
BYTE ***NumBlock***,
BYTE ***LgDiversifier***,
BYTE ***BlockDiversifier***,
BYTE ****StatusCard,***
WORD ****StatusSam,***
BYTE ****StatusChangeKey***)

Parameters :

I	BYTE	***NumKey***	Key Number (1 bytes)
I	BYTE	***VersionKeyA***	Key Version A (1 bytes)
I	BYTE	***VersionKeyB***	Key Version B (1 bytes)
I	LPBYTE	***DefaultAccess***	Default Access (4 bytes)
I	BYTE	***NumBlock***	Block number from 0 to 63 (1 bytes)
I	BYTE	***LgDiversifier***	Length Diversifier (1 bytes)
I	BYTE	***BlockDiversifier***	Block Diversifier (1 bytes)
O	BYTE	****StatusCard***	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK
O	BYTE	****StatusChangeKey***	Status Write (1 bytes) : 0x0A ➔ OK

MIFARE_ SAMNXP _Increment

Description : Increase a counter value.

Syntax : DWORD ***MIFARE_SAMNXP_Increment*** (BYTE ***NumBlock***,
LPBYTE ***Increment***,
BYTE ****StatusCard,***
WORD ****StatusSam,***
BYTE ****StatusIncrement***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter value from 1 to 63 (1 bytes)
I	LPBYTE	***Increment***	Value to add to the counter from 1 to 4294967295 (4 bytes)
O	BYTE	****StatusCard***	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK
O	BYTE	****StatusIncrement***	Status Write (1 bytes) : 0x0A ➔ OK

MIFARE_ SAMNXP _Decrement

Description : Decrease a counter value.

Syntax : DWORD ***MIFARE_SAMNXP_Decrement*** (BYTE ***NumBlock***,
LPBYTE ***Decrement***,
BYTE ****StatusCard,***
WORD ****StatusSam,***
BYTE ****StatusDecrement***)

Parameters :

I	BYTE	***NumBlock***	Block number of the counter value from 1 to 63 (1 bytes)
I	LPBYTE	***Decrement***	Value to subtract to the counter from 1 to 4294967295 (4 bytes)
O	BYTE	****StatusCard***	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK
O	BYTE	****StatusDecrement***	Status Write (1 bytes) : 0x0A ➔ OK

MIFARE_ SAMNXP _BackUpValue

Description : Copy the counter value in a backup location (the destination block must be in the same sector and configured also as a counter value).

Syntax : DWORD ***MIFARE_SAMNXP_BackupValue*** (BYTE ***Source***,
BYTE ***Destination,***
BYTE ****StatusCard,***
WORD ****StatusSam,***
BYTE ****StatusBackUp***)

Parameters :

I	BYTE	***Source***	Block number to copy
I	BYTE	***Destination***	Backup destination Block
O	BYTE	****StatusCard***	Status Card (1 bytes) : 0x00 ➔ OK
O	WORD	****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK
O	BYTE	**** StatusBackUp***	Status Write (1 bytes) : 0x0A ➔ OK

MIFARE_ SAMNXP _ResetAuthentication

Description : Disable a Mifare Card to forbid authenticated operation

Syntax : DWORD ***MIFARE_SAMNXP_ResetAuthentication*** (WORD ****StatusSam***)

Parameters :

O	WORD	****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK

1.13. MIFARE® PLUS card functions

Functions in this section allow managing all necessary operations on a MIFARE® PLUS card in Security Level 3. This class uses the NXP Mifare SAM AV2.

For all of these functions the error code table is listed here-under:
This function class is limited to the command available on the MIFARE® PLUS card.

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The received data are wrong
RCSC_CheckSum	CRC Error

MFP_SL3_Authentication

Description : Authenticate a given block.

Syntax : DWORD ***MFP_SL3_Authentication*** (BYTE ***SamKeyNum***,
BYTE ***SamKeyVersion***,
BYTE ***KeyBlockNum***,
BYTE ***LgDiversifier***,
LPBYTE ***Diversifier***,
BYTE ******StatusCard***,
WORD *****StatusSam)***

Parameters :

I	BYTE	***SamKeyNum***	Sam Key Number (1 bytes)
I	BYTE	***SamKeyVersion***	Sam Key Version (1 bytes)
I	BYTE	***KeyBlockNum***	Key Block Number - HigherByte, LowerByte (2 bytes)
I	BYTE	***LgDiversifier***	Length Diversifier (1 byte)
I	BYTE	***Diversifier***	Diversifier data (0 to 31 byte)
O	BYTE	******StatusCard***	Status Card (1 bytes) : 0x90 ➔ OK
O	WORD	*****StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK

MFP_SL3_ResetAuthentication

Description : Disable a MIFARE card to forbid authenticated operation.

Syntax : DWORD ***MFP_SL3_ResetAuthentication*** (BYTE ***Mode***,
BYTE **StatusCard*,
WORD **StatusSam)*

Parameters :

I	BYTE	***Mode***	Reset Mode (1 bytes)
O	BYTE	**StatusCard*	Status Card (1 bytes) : 0x90 ➔ OK
O	WORD	**StatusSam*	Status Sam (2 bytes) : 0x9000 ➔ OK

MFP_SL3_ReadBlock

Description : Read a block in a MIFARE card.

Syntax : DWORD ***MFP_SL3_ReadBlock*** (BYTE ***Mode***,
WORD ***BlockNum***,
WORD ***NumBlock***,
BYTE **StatusCard*,
WORD **StatusSam*,
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***Mode***	Read Mode (1 bytes)
I	WORD	***BlockNum***	Block Number to start reading (1 bytes)
I	WORD	***NumBlock***	Number of block to read (1 bytes)
O	BYTE	**StatusCard*	Status Card (1 bytes) : 0x90 ➔ OK
O	WORD	**StatusSam*	Status Sam (2 bytes) : 0x9000 ➔ OK
O	LPBYTE	***DataRead***	Data read from the card (0 - 240 bytes)

MFP_SL3_WriteBlock

Description : Write a block in a MIFARE card.

Syntax : DWORD ***MFP_SL3_WriteBlock*** (BYTE ***Mode***,
WORD ***BlockNum***,
WORD ***NumBlock***,
LPBYTE ***DataToWrite,***
BYTE ******StatusCard***,
WORD ******StatusSam***)

Parameters :

I	BYTE	***Mode***	Write Mode (1 bytes)
I	WORD	***BlockNum***	Block Number to start writing (1 bytes)
I	WORD	***NumBlock***	Number of block to write (1 bytes)
I	LPBYTE	***DataToWrite***	Data to Write in block (16 - 48 bytes)
O	BYTE	******StatusCard***	Status Card (1 bytes) : 0x90 ➔ OK
O	WORD	******StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK

MFP_SL3_ChangeKey

Description : Change a MIFARE Key in the card.

Syntax : DWORD ***MFP_SL3_ChangeKey*** (BYTE ***SamKeyNum***,
BYTE ***SamKeyVersion***,
WORD ***KeyBlockNum***,
BYTE ***LgDiversifier***,
LPBYTE ***Diversifier,***
BYTE ******StatusCard***,
WORD ******StatusSam***)

Parameters :

I	BYTE	***SamKeyNum***	Sam Key Number (1 bytes)
I	BYTE	***BlockNum***	Sam Key Version (1 bytes)
I	WORD	***NumBlock***	Key Block Number (2 bytes)
I	BYTE	***LgDiversifier***	Length Diversifier (1 bytes)
I	LPBYTE	***Diversifier***	Diversifier (0 to 31 bytes)
O	BYTE	******StatusCard***	Status Card (1 bytes) : 0x90 ➔ OK
O	WORD	******StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK

MFP_SL3_VirtualCardSupport

Description : Check Virtual Card is supported and retreive the UID.

Syntax : DWORD ***MFP_SL3_VirtualCardSupport*** (BYTE ***SamKeyNumVCENC***,
BYTE ***SamKeyVersionVCENC***,
BYTE ***SamKeyNumVCMAC***,
BYTE ***SamKeyVersionVCMAC***,
LPBYTE ***IID,***
BYTE ******StatusCard***,
WORD ******StatusSam,***
LPBYTE ***UID***)

Parameters :

I	BYTE	***SamKeyNumVCENC***	Sam Key Number for VC polling ENC (1 bytes)
I	BYTE	***SamKeyVersionVCENC***	Sam Key Version for VC polling ENC (1 bytes)
I	BYTE	***SamKeyNumVCMAC***	Sam Key Number for VC polling MAC (1 bytes)
I	BYTE	***SamKeyVersionVCMAC***	Sam Key Version for VC polling MAC (1 bytes)
I	LPBYTE	***IID***	Installation Identifier (16 byte)
O	BYTE	******StatusCard***	Status Card (1 bytes) : 0x90 ➔ OK
O	WORD	******StatusSam***	Status Sam (2 bytes) : 0x9000 ➔ OK
O	LPBYTE	***UID***	Real Card UID (4 - 7 bytes)

MFP_SL3_DeselectVirtualCard

Description : Deselect the Virtual Card.

Syntax : DWORD ***MFP_SL3_DeselectVirtualCard*** (BYTE ******StatusCard***)

Parameters :

O	BYTE	******StatusCard***	Status Card (1 bytes) : 0x90 ➔ OK

1.13. SRx Family card functions

Functions in this section allow managing all necessary operations on a SRx Family ticket (SR176, SR512, SR4K).

For all of these functions the error code table is listed here-under:
This function class is limited to the command available on the SRx Family ticket.

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The received data are wrong
RCSC_CheckSum	CRC Error

SRX_Active

Description : Activate and Select a SRx ticket.

Syntax : DWORD ***SRX_Active*** (BYTE *****Status***,
BYTE *****ChipType***,
LPBYTE ***UID***)

Parameters :

o	BYTE	*****Status***	Status (1 byte) : 0x0F ➔ OK
o	BYTE	** ***ChipType***	Type of ticket (1 byte)
o	LPBYTE	***UID***	UID from LSB to MSB (8 bytes)

SRX_ReadBlock

Description : Read Block.

Syntax : DWORD ***SRX_ReadBlock*** (BYTE ***BlockNum***,
BYTE ***NumBlock***,
BYTE ***ChipType***,
BYTE ****Lg***,
BYTE *****Status***,
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***BlockNum***	Block Number to start reading (1 bytes)
I	BYTE	***NumBlock***	Number of block to read (1 bytes)
I	BYTE	***ChipType***	Type of ticket (1 byte) 0 : SR176 1 : SR512 2 : SR4K
O	BYTE	**Lg***	Response Length (1 bytes)
O	BYTE	**Status***	Status (1 byte) : 0x02 ➔ OK
O	LPBYTE	***DataRead***	Data read from the card (n bytes)

SRX_WriteBlock

Description : Write and Verify Block.

Syntax : DWORD ***SRX_WriteBlock*** (BYTE ***BlockNum***,
BYTE ***NumBlock***,
LPBYTE ***DataToWrite,***
BYTE ***ChipType***,
BYTE ****Lg***,
BYTE **Status***,
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***BlockNum***	Block Number to start writing (1 bytes)
I	BYTE	***NumBlock***	Number of block to write (1 bytes)
I	LPBYTE	***DataToWrite***	Data to Write (n bytes)
I	BYTE	***ChipType***	Type of ticket (1 byte) 0 : SR176 1 : SR512 2 : SR4K
O	BYTE	**Lg***	Response Length (1 bytes)
O	BYTE	**Status***	Status (1 byte) : 0x02 ➔ OK
O	LPBYTE	***DataRead***	Data read from the card (n bytes)

SRX_Release

Description : Deactivate SRx ticket.

Syntax : DWORD ***SRX_Release*** (BYTE ***Param***,

BYTE *Status)

Parameters :

I	BYTE	***Param***	Deactivation of the ticket (1 byte) : 0x00 ➔ Deactivation
O	BYTE	****Status***	Status (1 byte) : 0x02 ➔ OK

SRX_Read

Description : Read Bytes at a given address.

Syntax : DWORD ***SRX_Read*** (WORD ***Add***,
BYTE ***NumBytes***,
BYTE ***ChipType***,
BYTE ***Lg***,
BYTE ****Status***,
LPBYTE ***DataRead***)

Parameters :

I	WORD	***Add***	Address of the first reading (2 bytes)
I	BYTE	***NumBytes***	Number of bytes to read (1 bytes)
I	BYTE	***ChipType***	Type of ticket (1 byte) 0 : SR176 1 : SR512 2 : SR4K
O	BYTE	****Lg***	Response Length (1 bytes)
O	BYTE	****Status***	Status (1 byte) : 0x02 ➔ OK
O	LPBYTE	***DataRead***	Data read from the card (n bytes)

SRX_Write

Description : Write and Verify Bytes at a given address.

Syntax : DWORD ***SRX_Write*** (WORD ***Add***,
BYTE ***NumBytes***,
LPBYTE ***DataToWrite,***
BYTE ***ChipType***,
BYTE ***Lg***,
BYTE ****Status***,
LPBYTE ***DataRead***)

Parameters :

I	WORD	***Add***	Address of the first reading ➔ (2 bytes)
I	BYTE	***NumBytes***	Number of bytes to read (1 bytes)
I	LPBYTE	***DataToWrite***	Data to Write (n bytes)
I	BYTE	***ChipType***	Type of ticket (1 byte) 0 : SR176 1 : SR512 2 : SR4K
O	BYTE	**Lg*	Response Length (1 bytes)
O	BYTE	**Status*	Status (1 byte) : 0x02 ➔ OK
O	LPBYTE	***DataRead***	Data read from the card (n bytes)

1.13. Desfire card functions

Functions in this section allow managing all necessary operations on a Desfire card.

For more information on functions of this section, see “DESFire Class” in the “RD-ST-08167-xx_ASK CSC - Coupler Software Interface_Gen5XX.pdf” document.

For all of these functions the error code table is listed here-under:
This function class is limited to the command available on the Desfire card.

Return : Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The received data are wrong
RCSC_CheckSum	CRC Error

DESFIRE Status

DESFire Card Response Data are of type SW1=0x91 SW2=0xYY

And DesFire Sam response Data are of type SW1=0x90 SW2=0xYY

Sw2(hex)	Status	Description
0x00	Operation ok	Successful operation
0x0C	No changes	No changes done to backup files, CommitTransaction / AbortTransaction not necessary
0x0E	Out of eeprom error	Insufficient Non Volatile memory to complete command
0x1C	Illegal command code	Command code not supported
0x1E	Integrity error	CRC or MAC does not match data
0x40	No such key	Invalid key number specified
0x7E	Length error	Length of command string invalid
0x9D	Permission denied	Current configuration/Status does not allow the requested command
0x9E	Parameter Error	Value of the parameter is invalid
0xA0	Application not found	Request AID not present on PICC
0xA1	Appli. Integrity error	Unrecoverable error within application, application will be disabled
0xAE	Authentication error	Current authentication status does not allow the requested command
0xAF	Additional frame	Additional data frame is expected to be sent
0xBE	Boundary error	Attempt to read/write data from/to beyond the file’s/record’s limits.
0xC1	PICC integrity error	Unrecoverable error within PICC, PICC will be disabled
0xCA	Command aborted	Previous command was not fully completed (not all frame were requested or provided)

0xCD	PICC disabled error	Picc was disabled by an unrecoverable error.
0xCE	Count error	Number of applications limited to 28, no additional Create Application possible.
0xDE	Duplicate error	Creation of file/application failed because file/application with same number already exists.
0xEA	No DesFireSam	No DesFire Sam is available.
0xEE	EEprom error	Could not complete NV write operation due to loss of power, internal backup/rollback mechanism activated
0xF0	File not found	Specified file number does not exist
0xF1	File integrity error	Unrecoverable error within file, file will be disabled
0xFC	Bad Length	Bad SAM response
0xFD	Bad Param	Bad parameter Applicative level the command is not transmitted to the card
0xFE	Bad Length	Bad length parameter Appli level (buffer overload … command is not given to the card)
0xFF	Timeout	No response from the card

DESFIRE_CreateApplication

Description : Create a new application in the card.

Syntax : DWORD ***DESFIRE_CreateApplication*** (LPBYTE ***AppID***,
BYTE ***Opt***,
BYTE ***KeyNum***,
WORD ****Status***)

Parameters :

I	LPBYTE	***AppID***	ID Number of the Appl in the card (3 byte)
I	BYTE	***Opt***	Options (1 byte) xxxx0001b Config changeable xxxx0010b Create/Delete operation are free (without master key) xxxx0100b Access to list directory is free (without master key) xxxx1000b master key setting can be changed
I	BYTE	***KeyNum***	Key Number usable for that new application (1 byte)
O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_DeleteApplication

Description : Deactivate application in the card.

Syntax : DWORD ***DESFIRE_DeleteApplication*** (LPBYTE ***AppID***,
WORD ****Status***)

Parameters :

I	LPBYTE	***AppID***	ID Number of the Appl in the card (3 byte)
O	WORD	***Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_SelectApplication

Description : Select one Application for further access in the card.

Syntax : DWORD ***DESFIRE_SelectApplication*** (LPBYTE ***AppID***,
WORD *****Status***)

Parameters :

I	LPBYTE	***AppID***	ID Number of the Appl in the card (3 byte)
O	WORD	***Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_FormatPICC

Description : Format card File system.

Syntax : DWORD ***DESFIRE_FormatPICC*** (WORD *****Status***)

Parameters :

O	WORD	***Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_GetApplicationIDs

Description : Retreive the current application ID.

Syntax : DWORD ***DESFIRE_GetApplicationIDs*** (BYTE ***NumID***,
BYTE *****Lg***,
WORD *****Status,***
LPBYTE ***IDs***)

Parameters :

I	BYTE	***NumID***	ID Number of the Appl in the card (3 byte)
O	BYTE	****Lg***	response length
O	WORD	***Status***	Status (2 byte) : 0x9100 ➔ OK
O	BYTE	***IDs***	ID for each application (n x 3 bytes)

DESFIRE_GetVersion

Description : Version of the card firmware.

Syntax : DWORD ***DESFIRE_GetVersion*** (WORD ****Status,***
LPBYTE ***HardInfo,***
LPBYTE ***SoftInfo,***
LPBYTE ***UID,***
LPBYTE ***Batch,***
BYTE ****Cw,***
BYTE ****Year***)

Parameters :

o	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK
o	LPBYTE	***HardInfo***	Hard Info (7 bytes) byte 1: code of the vendor byte 2; code of the type byte 3: code of the subtype byte 4: code of the major version number byte 5: code of the minor version number byte 6: code of the storage size
o	LPBYTE	***SoftInfo***	Soft Info (7 bytes) byte 1: code of the vendor byte 2; code of the type byte 3: code of the subtype byte 4: code of the major version number byte 5: code of the minor version number byte 6: code of the storage size byte 7: code of the communication protocol
o	LPBYTE	***UID***	Unique serial number (7 bytes)
o	LPBYTE	***Batch***	Production batch number (5 bytes)
o	BYTE	****Cw***	Calendar year of prod (1 byte)
o	BYTE	**** Year***	Year of manufacturing (1 byte)

DESFIRE_GetFreeMem

Description : retrieve the size available on the card.

Syntax : DWORD ***DESFIRE_GetFreeMem*** (WORD ****Status,***
LPBYTE ***Size***)

Parameters :

O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK
O	BYTE	***Size***	Size of free memory available (3 bytes)

DESFIRE_PrepareAuthentication

Description : This function sets parameters used for authentication.

Syntax : DWORD ***DESFIRE_PrepareAuthentication*** (BYTE ***AuthMode***,
BYTE ***SAMKeyNumber***,
BYTE ***SAMKeyVersion***,
WORD ****Status***)

Parameters :

I	BYTE	***AuthMode***	Authentication parameters (see RD_ST_08167-XX Coupler Software Interface GEN5XX or SAM AV2 specification).
I	BYTE	***SAMKeyNumber***	Key number in the SAM.
I	BYTE	***SAMKeyVersion***	Key version of the specified key in the SAM.
O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_Authenticate

Description : Perfoms the authentication.

Syntax : DWORD ***DESFIRE_Authenticate*** (BYTE ***KeyNum***,
WORD ****Status***)

Parameters :

I	BYTE	***KeyNum***	Number of the access key which will be used for the authentication (1 byte)
O	WORD	****Status***	Status (2 byte) : 0x9000 ➔ OK

DESFIRE_AuthenticateEV1

Description : This function authenticates a PICC or an application to grant access to files and to manage, if necessary, a secure communication. This new function (from firmware GEN5XX V1.21) replaces and enhances the “Prepare Authentication” and “Authenticate” commands. It allows 3 authentication functions of the DESFire EV1/EV2: ‘Authenticate’, ‘AuthenticateISO’ and ‘AuthenticateAES’..

Syntax :

DWORD ***DESFIRE_AuthenticateEV1*** (BYTE ***PICCKeyNumber***,
BYTE ***AuthMode***,
BYTE ***SAMKeyNumber***,
BYTE ***SAMKeyVersion***,
BYTE ***Type***,
BYTE ***LgDiversifier***,
BYTE *****Diversifier***,
WORD *****Status***)

Parameters :

I	BYTE	***PICCKeyNumber***	Specify the number of the access key which will be used for the authentication.
I	BYTE	***AuthMode***	Authentication parameters (see RD_ST_08167-XX Coupler Software Interface GEN5XX or SAM AV2 specification).
I	BYTE	***SAMKeyNumber***	Key number in the SAM.
I	BYTE	***SAMKeyVersion***	Key version of the specified key in the SAM.
I	BYTE	***Type***	Authentication type used. $00: TDEA DESFire 4 $01: TDEA ISO 10116 $02: AES
I	BYTE	***LgDiversifier***	length of the diversifier used for key diversification (0 if no diversification)
I	BYTE*	***Diversifier***	diversification data used for key diversification
O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_CommitTransaction

Description : Commits the transaction to end a transaction operation with changes.

Syntax : DWORD ***DESFIRE_CommitTransaction*** (WORD ****Status***)

Parameters :

O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_AbortTransaction

Description : Aborts the current transaction to end a transaction operation with no changes.

Syntax : DWORD ***DESFIRE_AbortTransaction*** (WORD ****Status***)

Parameters :

O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_ChangeKey

Description : This function allows changing any key stored on the PICC. If AID 00 00 00 is selected, the change applies to the Master key and therefore only the key N°00 is valid. This enhanced function is available from firmware GEN5XX V1.21.

Syntax :

DWORD ***DESFIRE_ChangeKey*** (BYTE ***CurKeyNo***,
BYTE ***CurKeyV***,
BYTE ***NewKeyNo***,
BYTE ***NewKeyV***,
BYTE ***KeyCompMeth***,
BYTE ***Cfg***,
BYTE ***Algo***,
BYTE ***LgDiversifier***,
BYTE ****Diversifier***,
WORD ****Status***)

Parameters :

I	BYTE	***CurKeyNo***	Current Key number in the SAM.
I	BYTE	***CurKeyV***	Current Key version in the SAM.
I	BYTE	***NewKeyNo***	New Key number in the SAM.
I	BYTE	***NewKeyV***	New Key version in the SAM.
I	BYTE	***KeyCompMeth***	Mask key compilation method. (See RD_ST_08167-XX Coupler Software Interface GEN5XX or SAM AV2 specification).
I	BYTE	***Cfg***	Key configuration bit 3...0: number of DESFire key to be changed bit 4: 1 if DESFire master key is to be changed.
I	BYTE	***Algo***	Algorithm used if PICC master key is changed bit 6...7: '00' specifies DES/2K3DES '01' specifies 3K3DES '10' specifies AES
I	BYTE	***LgDiversifier***	Length of the diversifier used for key diversification (0 if no diversification)
I	BYTE	****Diversifier***	Diversification data used for key diversification.
O	WORD	* ***Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_ChangeKeySetting

Description : Changes the key settings information.

Syntax : DWORD ***DESFIRE_ChangeKeySetting*** (BYTE ***KeySetting***,
WORD *__Status__)

Parameters :

I	BYTE	***KeySetting***	new master key settings either for the currently selected application or for the whole PICC (1 byte)
O	WORD	*__Status__	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_GetKeySetting

Description : Gets the configuration information on the PIDD and the application master key configuration settings.

Syntax : DWORD ***DESFIRE_GetKeySetting*** (WORD ***Status,***
BYTE ***KeySetting,***
BYTE ***NumKey***)

Parameters :

O	WORD	*__Status__	Status (2 byte) : 0x9100 ➔ OK
O	BYTE	* ***KeySetting***	key settings either for the currently selected application (1 byte)
O	BYTE	* ***NumKey***	Number of keys defined for the current selected application (1 byte)

DESFIRE_GetKeyVersion

Description : Gets Key Version.

Syntax : DWORD ***DESFIRE_GetKeyVersion*** (BYTE ***KeyNum***,
WORD ***Status,***
BYTE ***KeyVersion***)

Parameters :

I	BYTE	***KeyNum***	Specify the number of the access key (1 byte)
O	WORD	*__Status__	Status (2 byte) : 0x9100 ➔ OK

O	BYTE	* ***KeyVersion***	key Version (1 byte)

DESFIRE_ChangeFileSetting

Description : Changes the file configuration on the card.

Syntax : DWORD ***DESFIRE_ChangeFileSetting*** (BYTE ***FileID***,
BYTE ***CommEncrypted,***
BYTE ***CommMode,***
WORD ***AccessRight,***
WORD ****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file whose communication mode and access rights settings shall be changed (1 byte)
I	BYTE	***CommEncrypted***	Encrypt the communication (1 byte)
I	BYTE	***CommMode***	New communication mode (1 byte)
I	WORD	***AccessRight***	Specify the access right setting for this file (1 byte)
O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_ClearRecordFile

Description : Clears the record files selected by the input param.

Syntax : DWORD ***DESFIRE_ClearRecordFile*** (BYTE ***FileID***,
WORD ****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file which shall be cleared (1 byte)
O	WORD	****Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_CreateBackUpDataFile

Description : Creation of a Backup Data File.

Syntax : DWORD ***DESFIRE_CreateBackUpDataFile*** (BYTE ***FileID***,
BYTE ***CommMode,***

WORD ***AccessRight,***
LPBYTE ***FileSize***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the new Backup File is to be created (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***AccessRight***	New File access rights settings (2 byte)
I	LPBYTE	***FileSize***	Size of the new Backup File in bytes (3 byte)
O	WORD	***Status**	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_CreateCyclicRecordFile

Description : Creation of a Cyclic Data File.

Syntax : DWORD ***DESFIRE_CreateCyclicRecordFile*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***AccessRight,***
LPBYTE ***RecordSize***,
LPBYTE ***MaxNumRecord***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the Cyclic record is to be created (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***AccessRight***	New File access rights settings (2 byte)
I	LPBYTE	***RecordSize***	Size of the new Cyclic File in bytes (3 byte)
I	LPBYTE	***MaxNumRecord***	Number of the records for the new Cyclic File (3 byte)
O	WORD	***Status**	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_CreateLinearRecordFile

Description : Creation of a Linear Data File.

Syntax : DWORD ***DESFIRE_CreateLinearRecordFile*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***AccessRight,***
LPBYTE ***RecordSize***,
LPBYTE ***MaxNumRecord***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the new Linear record File is to be created (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***AccessRight***	New File access rights settings (2 byte)
I	LPBYTE	***RecordSize***	Size of the new linear File in bytes (3 byte)
I	LPBYTE	***MaxNumRecord***	Number of the records for the new linear File (3 byte)
O	WORD	**Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_CreateStandardDataFile

Description : Creation of a Standard Data File.

Syntax : DWORD ***DESFIRE_CreateStandardDataFile*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***AccessRight,***
LPBYTE ***FileSize***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the new File is to be created (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***AccessRight***	New File access rights settings (2 byte)
I	LPBYTE	***FileSize***	Size of the new File in bytes (3 byte)
O	WORD	**Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_CreateValueFile

Description : Creation of a Value File.

Syntax : DWORD ***DESFIRE_CreateValueFile*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***AccessRight,***
LPBYTE ***Lower***,
LPBYTE ***Upper***,
LPBYTE ***Initial,***
BYTE ***Limited,***
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the new File is to be created (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***AccessRight***	New File access rights settings (2 byte)
I	LPBYTE	***Lower***	Min amount for the value file (4 byte)
I	LPBYTE	***Upper***	Max amount for the value file (4 byte)
I	BYTE	***Initial***	Amount with which the value file will be created (4 byte)
I	LPBYTE	***Limited***	Limited credit command is enabled for the new value file (1 byte)
O	WORD	***Status**	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_Credit

Description : Credit a Value on a Value File.

Syntax : DWORD ***DESFIRE_Credit*** (BYTE ***FileID***,
BYTE ***CommMode,***
LPBYTE ***Amount***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the new File is to be credited (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	LPBYTE	***Amount***	Amount to be credited in the value file (4 byte)
O	WORD	***Status**	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_Debit

Description : Debit a Value on a Value File.

Syntax : DWORD ***DESFIRE_Debit*** (BYTE ***FileID***,
BYTE ***CommMode,***
LPBYTE ***Amount***,
WORD *__Status__)

Parameters :

I	BYTE	***FileID***	ID of the file for which the new File is to be debited (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	LPBYTE	***Amount***	Amount to be debited in the value file (4 byte)
O	WORD	*__Status__	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_DeleteFile

Description : Delete a Value File.

Syntax : DWORD ***DESFIRE_DeleteFile*** (BYTE ***FileID***,
WORD *__Status__)

Parameters :

I	BYTE	***FileID***	ID of the file for which the new File is to be deleted (1 byte)
O	WORD	*__Status__	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_GetFileID

Description : Get File ID for the current application.

Syntax : DWORD ***DESFIRE_GetFileID*** (BYTE ***MaxFileID***,
WORD *__Status,__
BYTE *__NbFound__,
LPBYTE ***FileId***)

Parameters :

I	BYTE	***MaxFileID***	Max response expected (1 byte)

O	WORD	*Status	Status (2 byte) : 0x9100 ➔ OK
O	BYTE	* NbFound	Number of FileId found “n” (1 byte)
O	LPBYTE	FileId	FileID array (n bytes)

DESFIRE_GetFileSetting

Description : Get File ID for the current application.

Syntax : DWORD ***DESFIRE_GetFileSetting*** (BYTE ***FileID***,
WORD ****Status,***
BYTE ****FileType***,
BYTE ****CommMode,***
WORD ****AccessRight***)

Parameters :

I	BYTE	FileID	ID of the file for which the setting is to be Retrieve (1 byte)
O	WORD	*Status	Status (2 byte) : 0x9100 ➔ OK
O	BYTE	*FileType	Type of File (1 byte)
O	BYTE	*CommMode	File communication mode (1 byte)
O	WORD	* AccessRight	File access rights settings (2 bytes)

DESFIRE_GetValue

Description : Get File Settings for the current application.

Syntax : DWORD ***DESFIRE_GetValue*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ****Status,***
LPBYTE ***Amount***)

Parameters :

I	BYTE	FileID	ID of the file for which the setting is to be Retrieve (1 byte)
I	BYTE	CommMode	File communication mode (1 byte)
O	WORD	*Status	Status (2 byte) : 0x9100 ➔ OK
O	LPBYTE	Amount	Amount of the value returned (4 bytes)

DESFIRE_LimitedCredit

Description : Limited Credit.

Syntax : DWORD ***DESFIRE_LimitedCredit*** (BYTE ***FileID***,
BYTE ***CommMode,***
LPBYTE ***Amount***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the credit is to increase (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	LPBYTE	***Amount***	Max Amount that can be added (4 bytes)
O	WORD	***Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_ReadData

Description : Read Data standard File.

Syntax : DWORD ***DESFIRE_ReadData*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***FromOffset,***
WORD ***NumByteToRead,***
WORD *****Status,***
WORD *****NumByteRead,***
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the setting is to be Retrieve (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***FromOffset***	Offset in the File (2 bytes)
I	WORD	***NumByteToRead***	Nb byte to read (2 bytes)
O	WORD	***Status***	Status (2 byte) : 0x9100 ➔ OK
O	WORD	***NumByteRead***	Nb Bytes read "n" (2 bytes)
O	LPBYTE	***DataRead***	Data read in the File (n bytes)

DESFIRE_WriteData

Description : WriteData standard File.

Syntax : DWORD ***DESFIRE_WriteData*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***FromOffset,***
WORD ***NumByteToWrite,***
LPBYTE ***DataToWrite***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the setting is to be Retrieve (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***FromOffset***	Offset in the File (2 bytes)
I	WORD	***NumByteToWrite***	Nb byte to write (2 bytes)
I	LPBYTE	***DataToWrite***	Data write in the File (n bytes)
O	WORD	**Status***	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_ReadRecord

Description : Read Data Record File.

Syntax : DWORD ***DESFIRE_ReadRecord*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***FromRecord,***
WORD ***NumRecordToRead,***
WORD ***RecordSize,***
WORD *****Status,***
WORD *****NumRecordRead,***
LPBYTE ***DataRead***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the setting is to be Retrieve (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***FromOffset***	Offset in the File (2 bytes)
I	WORD	***NumRecordToRead***	Number of record to read (2 bytes)

I	WORD	***RecordSize***	Record size (2 bytes)
O	WORD	**Status*	Status (2 byte) : 0x9100 ➔ OK
O	WORD	**NumRecordRead*	Nb Bytes read "n" (2 bytes)
O	LPBYTE	***DataRead***	Data read in the File (n bytes)

DESFIRE_WriteRecord

Description : WriteData record File.

Syntax : DWORD ***DESFIRE_WriteRecord*** (BYTE ***FileID***,
BYTE ***CommMode,***
WORD ***FromOffset,***
WORD ***NbDataToToWrite,***
LPBYTE ***DataToWrite***,
WORD *****Status***)

Parameters :

I	BYTE	***FileID***	ID of the file for which the setting is to be Retrieve (1 byte)
I	BYTE	***CommMode***	File communication mode (1 byte)
I	WORD	***FromOffset***	Offset in the File (2 bytes)
I	WORD	***NbDataToWrite***	Number of data to write (2 bytes)
I	LPBYTE	***DataToWrite***	Data write in the File (n bytes)
O	WORD	**Status*	Status (2 byte) : 0x9100 ➔ OK

DESFIRE_SamGetVersion

Description : Sam Firmware Info.

Syntax : DWORD ***DESFIRE_SamGetVersion*** (WORD *****Status,***
LPBYTE ***SamVersion***)

Parameters :

O	WORD	**Status*	Status (2 byte) : 0x9000 ➔ OK
O	LPBYTE	***SamVersion***	Version SAM (32 bytes)

DESFIRE_SamSelectApplication

Description : Select an application in the SAM.

Syntax : DWORD ***DESFIRE_SamSelectApplication*** (LPBYTE ***DirFileAID***, WORD *****Status***)

Parameters :

I	LPBYTE	***DirFileAID***	Directory File AID (3 bytes)
O	WORD	***Status**	Status (2 byte) : 0x9000 ➔ OK

DESFIRE_SamLoadInitVector

Description : Load an init vector in the SAM for 3DES seeding.

Syntax : DWORD ***DESFIRE_SamLoadInitVector*** (LPBYTE ***InitVector***, WORD *****Status***)

Parameters :

I	LPBYTE	***InitVector***	Crypto seed (8 bytes)
O	WORD	***Status**	Status (2 byte) : 0x9000 ➔ OK

DESFIRE_SamGetKeyEntry

Description : Get Key entry Info.

Syntax : DWORD ***DESFIRE_SamGetKeyEntry*** (BYTE ***KeyNum*,** BYTE ***Lg*,** WORD ***Status*,** LPBYTE ***KeyEntry***)

Parameters :

I	BYTE	***KeyNum***	Key Entry Number (1 bytes)
O	BYTE	***Lg***	Length response (1 byte)
O	WORD	***Status**	Status (2 byte) : 0x9000 ➔ OK
O	LPBYTE	***KeyEntry***	3 Key Versions (3 bytes)

DESFIRE_SamGetKucEntry

Description : Get Key Usage Counter Info.

Syntax : DWORD ***DESFIRE_SamGetKucEntry*** (BYTE ***RefKucNum*** ,
BYTE *****Lg*** ,
WORD *****Status*** ,
LPBYTE ***KucEntry***)

Parameters :

I	BYTE	***RefKucNum***	Key Usage Counter Entry Reference Number (1 bytes)
O	BYTE	****Lg***	Length response (1 byte)
O	WORD	****Status***	Status (2 byte) : 0x9000 ➔ OK
O	LPBYTE	***KucEntry***	Key Usage Counter Versions (n bytes)

DESFIRE_SamDisableCrypto

Description : Disable the crypto of certain function on the SAM/PICC.

Syntax : DWORD ***DESFIRE_SamDisableCrypto*** (WORD ***PROMAS*** ,
WORD *****Status***)

Parameters :

I	WORD	***PROMAS***	Programming bit Mask (2 bytes)
O	WORD	****Status***	Status (2 byte) : 0x9000 ➔ OK

1.13. Mifare Ultralight C and Mifare Ultralight EV1 functions

Functions in this section allow managing of Mifare UltraLight C and Mifare UltraLight EV1.

Although the Mifare Ultralight (not Ultralight C neither Ultralight EV1) is already managed by the CTX512x functions, these functions can also be used.

For more information on functions of this section, see “Mifare UltraLight C and Mifare UltraLight EV1 Class” in the “RD-ST-08167-xx_ASK CSC - Coupler Software Interface_Gen5XX.pdf” document.

For all of these functions the error code table is listed here-under:

Return: Return value

RCSC_Ok	The function succeeds
RCSC_OpenCOMError	The PC communication port opening fails
RCSC_Timeout	Timeout coupler
RCSC_Fail	The function fails
RCSC_DataWrong	The received data are wrong
RCSC_CheckSum	CRC Error

MFUL_Identify

Description : Determines the Mifare UltraLight type. This command performs card detection. This command is not mandatory. However, this command will allow strict parameters checking for other functions of this section. After this command on Mifare ULC, the card must be detected using CSC_SearchCardExt function with ISOA type, before using others MFUL functions.

Syntax : DWORD ***MFUL_Identify*** (BYTE ***RFU***, BYTE *****Status***)

Parameters :

I	BYTE	***RFU***	RFU, should be set to 0.
O	BYTE	***Status**	$00 no answer $01 bad CRC $10 + NAK code from MFUL (see NAK codes) $20 Mifare UltraLight (MF0ICU1) $21 Mifare UltraLight C (MF0ICU2 $22 Mifare UltraLight EV1 640 bits (MF0UL11) $23 Mifare UltraLight EV1 1312 bits (MF0UL21) $24 unknown Mifare UltraLight $25 another ISO14443A chip

MFUL_Read

Description : Reads of a number of bytes at a given address

Syntax : DWORD ***MFUL_Read*** (BYTE ***ByteAddress***, BYTE ***Nb,*** BYTE *****Status***
BYTE *****LngData,*** BYTE *****ReadData***)

Parameters :

I	BYTE	***ByteAddress***	Address of the first byte to read, multiple of 4. 0…$3C for Mifare UltraLight (MF0ICU1) 0…$AC for Mifare UltraLight C (MF0ICU2) 0…$4C for Mifare UltraLight EV1 640 bits (MF0UL11) 0…$A0 for Mifare UltraLight EV1 1312 bits (MF0UL21)
I	BYTE	***Nb***	Number of bytes to read 0…$40 for Mifare UltraLight (MF0ICU1) 0…$B0 for Mifare UltraLight C (MF0ICU2) 0…$50 for Mifare UltraLight EV1 640 bits (MF0UL11) 0…$A4 for Mifare UltraLight EV1 1312 bits (MF0UL21)
O	BYTE	**Status*	$00 No answer $01 Bad CRC $02 Success $03 Bad Parameters $10 + NAK code from MFUL (see NAK codes) Note: if Status is different from $02 or $03, the card will come into the HALT state, so CSC_SearchCardExt function should be called to perform other operation.
O	BYTE	**LngData*	Read data length
O	BYTE	**ReadData*	Read data

MFUL_Write

Description : Writes, then checks by reading the bytes written at a given address

Syntax : DWORD ***MFUL_Write*** (BYTE ***ByteAddress***, BYTE ***Nb,***
BYTE *****DataToWrite,*** BYTE *****Status***
BYTE *****LngData,*** BYTE *****ReadData***)

Parameters :

I	BYTE	***ByteAddress***	address of the first byte to write, multiple of 4 0…$3C for Mifare UltraLight (MF0ICU1) 0…$BC for Mifare UltraLight C (MF0ICU2) 0…$4C for Mifare UltraLight EV1 640 bits (MF0UL11) 0…$A0 for Mifare UltraLight EV1 1312 bits (MF0UL21)
I	BYTE	***Nb***	number of bytes to write, multiple of 4 0…$40 for Mifare UltraLight (MF0ICU1) 0…$C0 for Mifare UltraLight C (MF0ICU2) 0…$50 for Mifare UltraLight EV1 640 bits (MF0UL11) 0…$A4 for Mifare UltraLight EV1 1312 bits (MF0UL21)
I	BYTE	****DataToWrite***	data to write
O	BYTE	**Status*	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes) $82 Verification failure; read data are returned. Note: if Status is different from $02 or $03, the card will come into the HALT state, so CSC_SearchCardExt function should be called to perform other operation.
O	BYTE	**LngData*	Read data length
O	BYTE	**ReadData*	Read data

MFULC_Authenticate

Description : Performs mutual authentication, to access protected area. This function uses a NXP SAM AV2. After power up or coupler reset, the SAM must be reset before using this function.

Syntax : DWORD ***MFULC_Authenticate*** (BYTE ***KeyNo***, BYTE ***KeyV,***
BYTE ***DIVLength,*** BYTE *****DIVInput,***
BYTE *****Status,*** WORD *****SAMStatus***)

Parameters :

I	BYTE	***KeyNo***	key reference number of key entry ($00 to $7F)
I	BYTE	***KeyV***	key version of KeyNo ($00 to $FF)
I	BYTE	***DIVLength***	length of the diversification input (0 to 31, 0 = no diversification)
I	BYTE	****DIVInput***	diversification input
O	BYTE	*** Status***	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes)
O	WORD	*** SAMStatus***	$90 00 correct execution, authentication successful $90 1E correct execution, authentication failed Other execution not correct, see Mifare SAM AV2 (P5DF081) datasheet.

MFULC_WriteKeyFromSAM

Description : Retrieves the key from the NXP AV2 SAM and writes it in the Mifare UltraLight C.
The key can also be written directly by the application, using the MFUL_Write function.
Note: as the SAM key entry should be dumpable, this key should be only present on SAMs in protected personalizing/issuing machines.
This function uses a NXP SAM AV2. After power up or coupler reset, the SAM must be reset before using this function.

Syntax : DWORD ***MFULC_WriteKeyFromSAM*** (BYTE ***KeyNo***, BYTE ***KeyV,***
BYTE ***DIVLength,*** BYTE *****DIVInput,***
BYTE *****Status,*** WORD *****SAMStatus***)

Parameters :

I	BYTE	***KeyNo***	key reference number of key entry ($00 to $7F)
I	BYTE	***KeyV***	key version of KeyNo ($00 to $FF)
I	BYTE	***DIVLength***	length of the diversification input (0 to 31, 0 = no diversification)
I	BYTE	****DIVInput***	diversification input
O	BYTE	****Status***	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes)
O	WORD	****SAMStatus***	$90 00 correct execution, authentication successful $90 1E correct execution, authentication failed Other execution not correct, see Mifare SAM AV2 (P5DF081) datasheet.

MFULEV1_PasswordAuthenticate

Description : Performs password authentication, to access protected area.

Syntax : DWORD ***MFULEV1_PasswordAuthenticate*** (BYTE ****Password,*** BYTE ****Status,*** BYTE ****PACK***)

Parameters :

I	BYTE	***Password**	password value for authentication (4 bytes)
O	BYTE	***Status**	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes)
O	BYTE	***PACK**	Password Authentication Acknowledge (2 bytes, this is the value from the memory, PACK area)

MFULEV1_CreateDiversifiedPasswordandPACK

Description : Create a diversified password and password acknowledge from SAM. Can be used before to personalize PWD and PACK, and before to use the MFULEV1_PasswordAuthenticate function. This function is not mandatory. It helps the application to create a diversified password. It does not write the password to the MFUL EV1. The application should write the password and PACK values in the memory using the MFUL_Write function. This function uses a NXP SAM AV2. After power up or coupler reset, the SAM must be reset before using this function.

Syntax : DWORD ***MFULEV1_CreateDiversifiedPasswordandPACK*** (

BYTE ***KeyNo***, BYTE ***KeyV,***

BYTE ***DIVLength,*** BYTE *****DIVInput,***

WORD *****SAMStatus,***

BYTE *****Password,*** BYTE *****PACK***)

Parameters :

I	BYTE	***KeyNo***	key reference number of key entry ($00 to $7F)
I	BYTE	***KeyV***	key version of KeyNo ($00 to $FF)
I	BYTE	***DIVLength***	length of the diversification input (1 to 31)
I	BYTE	****DIVInput***	diversification input
O	WORD	****SAMStatus***	$90 00 correct execution, authentication successful $90 1E correct execution, authentication failed Other execution not correct, see Mifare SAM AV2 (P5DF081) datasheet.
O	BYTE	****Password***	Diversified password (4 bytes)
O	BYTE	****PACK***	Diversified Password Authentication Acknowledge (2 bytes)

MFULEV1_ReadCounter

Description : Reads the current value of one of the 3 one-way counters.

Syntax : DWORD ***MFULEV1_ReadCounter*** (BYTE ***CounterNb,*** BYTE ****Status,***
DWORD ****CounterValue***)

Parameters :

I	BYTE	***CounterNb***	counter number from $00 to $02
O	BYTE	****Status***	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes)
O	DWORD	****CounterValue***	counter value from $000000 to $FFFFFF

MFULEV1_IncrementCounter

Description : Increments one of the 3 one-way counters.

Syntax : DWORD ***MFULEV1_IncrementCounter*** (BYTE ***CounterNb,***
DWORD ***IncrementValue,***
BYTE ****Status***)

Parameters :

I	BYTE	***CounterNb***	counter number from $00 to $02
I	DWORD	***IncrementValue***	increment value from $000000 to $FFFFFF
O	BYTE	****Status***	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes)

MFULEV1_GetVersion

Description : Retrieves information about the Mifare UltraLight EV1.

Syntax : DWORD ***MFULEV1_GetVersion*** (BYTE ****Status,***
BYTE ****LngData,*** BYTE ****Data***)

Parameters :

o	BYTE	****Status***	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes)
o	BYTE	****LngData***	Read data length (8)
o	BYTE	****ReadData***	Version Information, 8 bytes, see below

Version Information:

Byte no.	Description	MF0UL11	MF0UL21	Interpretation
0	fixed header	00h	00h	
1	vendor ID	04h	04h	NXP Semiconductors
2	product type	03h	03h	MIFARE UltraLight
3	product subtype	01h	01h	17 pF
4	major product version	01h	01h	EV1
5	minor product version	00h	00h	V0
6	storage size	0Bh	0Eh	0B=between 32 and 64 bytes (48) 0E=128 bytes
7	protocol type	03h	03h	ISO/IEC 14443-3 compliant

MFULEV1_CheckTearingEvent

Description : Identifies if a tearing event happened on a specified counter.

Syntax : DWORD ***MFULEV1_CheckTearingEvent*** (BYTE ***CounterNb***,
BYTE *****Status,***
BYTE *****Valid***)

Parameters :

I	BYTE	***CounterNb***	counter number from $00 to $02
O	BYTE	*****Status***	$00 No answer $01 Bad CRC $02 Success $03 Bad parameters $10 + NAK code from MFUL (see NAK codes)
O	BYTE	*****Valid***	valid flag, $BD for normal operation, otherwise a tearing event has happened.

2. DLL FUNCTIONS USE EXAMPLE

```
/*-------------------------------------------------------------------------
--- Demo program for CT2000
-------------------------------------------------------------------------*/

#include  <windows.h>                                   // windows library
#include  <stdio.h>                                     // input/output library
#include  "askcsc.h"                                    // ASK-csc library


#define             VERSIONLENGTH 80                    // length of version
#define             RESULTLENGHT  10                    // close session result buffer length
#define             SUCCESS       0x01                  // function succeeds
#define             FAILURE       0x00                  // function fails
#define             STATUS_OK     {0x00, 0x90, 0x00}    // reader returned status OK
#define             FORG          0x01                  // forget (or not) last serial number
#define             TIMEOUT       0                     // (no) time out when looking for a card
#define             REC_LENGTH    0x1D                  // length of the recordings
#define             READ_REC      0x00                  // record read (or not) when opening session

#define             DEFAULT       0x00                  // acces mode "default"
#define             PROTECTED     0x01                  // acces mode "protected"
#define             STAMPED       0x02                  // acces mode "stamped"
#define             PERSO         0x00                  // session mode = "perso"
#define             RELOAD        0x01                  // session mode = "reload"
#define             VALID         0x02                  // session mode = "valid"

// addresses of CT2000 keys in SAM
#define             MF_PER_KEY    0x10
#define             MF_PAR_KEY    0x12
#define             MF_INV_KEY    0x13
#define             MF_STR_KEY    0x14


#define             OLD_PIN       {0x30,0x30,0x30,0x30}       // old PIN number
#define             NEW_PIN       {0x30,0x30,0x30,0x30}       // new PIN number

#define             DATA_TO_WRITE {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,
                    0x0c,0x0d,0x0e,0x0f,0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d}



/*-------------------------------------------------------------------------
--- Function statement
-------------------------------------------------------------------------*/

static DWORD checkSuccess(DWORD ret, sCARD_Status status);



/*-------------------------------------------------------------------------
--- Main
-------------------------------------------------------------------------*/

void main(void)
{
        sCARD_Status    status;                         // reader status
        sCARD_Session   session;                        // application data return value
        sCARD_SecurParam        securParam;             // security parameters
        sCARD_SearchExt cardSearch;                     // search structure
        DWORD           ret;                            // functions returned value
        BYTE            result[RESULTLENGHT];           // session closure order result
        DWORD           cbResult;                       // length of order result in session closure
        BYTE            oldPin[]=OLD_PIN;               // old pin number in card
        BYTE            newPin[]=NEW_PIN;               // new pin number in card
        BYTE            data[REC_LENGTH];               // data buffer (reading purpose)
        BYTE            dataToWrite[REC_LENGTH]=DATA_TO_WRITE;          // example data written
        BYTE            indice=0;                       // loop counter
        char            version[VERSIONLENGTH];         // buffer for SW version
        DWORD           search_mask;                    // search mask
        BYTE            com;                            // comm mode found
        DWORD           respLength;                     // response length
        BYTE            response[29]="";                // response of the card
```

```
//--- search init
cardSearch.CONT=0x00;
cardSearch.INNO=0x01;
cardSearch.ISOA=0x00;
cardSearch.ISOB=0x00;
cardSearch.MIFARE=0x00;
cardSearch.MV4k=0x00;
cardSearch.MV5k=0x00;
cardSearch.TICK=0x00;
search_mask=SEARCH_MASK_INNO;

//--- open COM
ret=CSC_SearchCSC();
if (ret!=RCSC_Ok)
{
	printf("error when opening communication\n");
}
else
{
	ret=CSC_VersionCSC(version);			// display SW version
	if (ret!=RCSC_Ok)
	{
		printf("error when reading version\n");
	}
	else
	{
		printf("version :\n%s\n\n",version);	// print SW version


		//---- change PIN ---------------------------------------------

		securParam.AccMode=DEFAULT;			// acces mode
		securParam.NKEY=MF_PER_KEY;			// key number to use
		securParam.SID=0x00;				// short ID of the MF
		securParam.LID=0x3F00;				 // long ID of the MF
		securParam.RFU=0x00;				 // reserved for future use

		printf("\n------change PIN section------\n");
		printf("'perso mode' - key used : 0x%x\n", securParam.NKEY);

		fflush(stdin);
		getchar();

		// active CT2000
		ret=CSC_SearchCardExt(&cardSearch,search_mask,FORG,TIMEOUT,&com,&respLength,response);
		if ((ret!=RCSC_Ok)||(com!=0x03))
		{
			printf("no CT2000 found\n");
		}
		else
		{
			printf("CT2000 found\n---\nchanging PIN\n");

			// change PIN with MF_PER_KEY
			ret=ChangePIN(securParam, oldPin, newPin, &status);
			if (checkSuccess(ret, status)!=SUCCESS)
			{
				printf("!failed in changing PIN\n");
			}
			else
			{
				printf("PIN changed successfully\n");
			}
			printf("---\nclosing com with the card.....");
```

```
		// close COM with CT2000
		ret=CSC_CardEnd();
		if (ret!=RCSC_Ok)
		{
			printf("!failed in closing com with the card\n");
		}
		else
		{
			printf("com closed successfully\n");
		}
}


//---- update ID file (EF) ------------------------------------
//     in reload session

securParam.AccMode=DEFAULT;
securParam.NKEY=MF_PAR_KEY;
securParam.SID=0x03;		// ID file selected
securParam.LID=0x0003;
securParam.RFU=0x00;

printf("\n------update ID file section------\n");
printf("session mode - reload - key used : 0x%x\n", securParam.NKEY);

fflush(stdin);
getchar();

// active CT2000
ret=CSC_SearchCardExt(&cardSearch,search_mask,FORG,TIMEOUT, &com,&respLength,response);
if ((ret!=RCSC_Ok)||(com!=0x03))
{
	printf("no CT2000 found\n");
}
else
{

		// open reload session
		ret=OpenSession(RELOAD, securParam, READ_REC,&session, &status);
		if (checkSuccess(ret, status)!=SUCCESS)
		{
			printf("!failed in opening session\n");
		}
		else
		{
			printf("session opened successfully\n");
			printf("CT2000 found\n---\nupdating ID EF\n");


			// update record session mode, reload
			ret=UpdateRecord(securParam, 0x01,REC_LENGTH, dataToWrite, &status);
			if (checkSuccess(ret, status)!=SUCCESS)
			{
				printf("!failed in updating ID EF\n");
			}
			else
			{
				printf("ID EF updated successfully\n");
				printf("---\nclosing session\n");

				// close reload session
				ret=CloseSession(result, &cbResult, &status);
				if (checkSuccess(ret, status)!=SUCCESS)
				{
					printf("!failed in closing session\n");
				}
				else
				{
					printf("session closed successfully\n");
				}
			}
		}
		printf("---\nclosing com with the card.....");
```

```
		// close COM with CT2000
		ret=CSC_CardEnd();
		if (ret!=RCSC_Ok)
		{
			printf("!failed in closing com with the card\n");
		}
		else
		{
			printf("com closed successfully\n");
		}
}

//---- read ID file (EF) -------------------------------------
//     PIN mode

securParam.AccMode=DEFAULT;
securParam.NKEY=MF_INV_KEY;
securParam.SID=0x03;		// ID file selected
securParam.LID=0x0003;
securParam.RFU=0x00;

printf("\n------read ID file section------\n");
printf("'PIN' mode - key used : 0x%x\n", securParam.NKEY);

fflush(stdin);
getchar();

// active CT2000
ret=CSC_SearchCardExt(&cardSearch,search_mask,FORG,TIMEOUT, &com,&respLength,response);
if ((ret!=RCSC_Ok)||(com!=0x03))
{
	printf("no CT2000 found\n");
}
else
{
	printf("CT2000 found\n");
	printf("---\nverifying PIN\n");

	// verify PIN first
	ret=VerifyPIN(securParam, newPin, &status);
	if (checkSuccess(ret, status)!=SUCCESS)
	{
		printf("!failed in verifying PIN\n");
	}
	else
	{
		printf("PIN verified successfully\n");
		printf("---\nreading data\n");

		// read data (enabled by PIN verif)
		ret=ReadRecord(securParam, 0x01, REC_LENGTH,
			data, &status);
		if (checkSuccess(ret, status)!=SUCCESS)
		{
			printf("!failed in reading ID file\n");
		}
		else
		{
			printf("data read :(0x) ");
			for (indice=0; indice<REC_LENGTH; indice++)
			{
				printf("%x ", data[indice]);
			}
			printf("\ndata read successfully\n");
		}
	}
	printf("---\nclosing com with the card.....");
```

```
                    // close COM with CT2000
                    ret=CSC_CardEnd();
                    if (ret!=RCSC_Ok)
                    {
                            printf("!failed in closing com with the card\n");
                    }
                    else
                    {
                            printf("com closed successfully\n");
                    }
            }

            //---- read ICC file (EF) ------------------------------------
            //     stamped mode

            securParam.AccMode=STAMPED;
            securParam.NKEY=MF_STR_KEY;
            securParam.SID=0x02;            // ICC file selected
            securParam.LID=0x0002;
            securParam.RFU=0x00;

            printf("\n------read ICC file section------\n");
            printf("'STAMPED' mode - key used : 0x%x\n", securParam.NKEY);

            fflush(stdin);
            getchar();

            // active CT2000
            ret=CSC_SearchCardExt(&cardSearch,search_mask,FORG,TIMEOUT,&com,&respLength,response);
            if ((ret!=RCSC_Ok)||(com!=0x03))
            {
                    printf("no CT2000 found\n");
            }
            else
            {
                    printf("CT2000 found\n--\nreading ICC EF\n");

                    // read record-stamped mode
                    ret=ReadRecord(securParam, 0x01, REC_LENGTH,data, &status);
                    if (checkSuccess(ret, status)!=SUCCESS)
                    {
                            printf("!failed in reading ICC file\n");
                    }
                    else
                    {
                            printf("data read :(0x) ");
                            for (indice=0; indice<REC_LENGTH; indice++)
                            {
                                    printf("%x ", data[indice]);
                            }
                            printf("\ndata read successfully\n");
                    }
                    printf("---\nclosing com with the card.....");

                    // close COM with CT2000
                    ret=CSC_CardEnd();
                    if (ret!=RCSC_Ok)
                    {
                            printf("!failed in closing com with the card\n");
                    }
                    else
                    {
                            printf("com closed successfully\n");
                    }
            }
        }
    }
}
```

```
/*-------------------------------------------------------------------------------
--- Function description
-------------------------------------------------------------------------------*/


/******************************************************************************
checkSucces function

  Purpose : checks both the function an the reader status

            Input     : ret       :       status returned by the function (RCSC_Ok or other)
                      : status    :       execution status of the reader ({00, 90, 00} or other)
            Output    : void
            Return    : SUCCESS / FAILURE
            other     : prints function and readers status
******************************************************************************/
static DWORD checkSuccess(DWORD ret, sCARD_Status status)
/******************************************************************************/
{

            sCARD_Status      statusOk = STATUS_OK;     // satus OK
            DWORD                       retu;                     // returned value

            if ((ret==RCSC_Ok) &&
                        (status.Code==statusOk.Code ) &&
                        (status.Byte1==statusOk.Byte1 ) &&
                        (status.Byte2==statusOk.Byte2 ))
            {
                        retu=SUCCESS;
            }
            else
            {
                        retu=FAILURE;
            }
            printf("function status : %4x , reader status : {%x,%x,%x}\n",
                        ret, status.Code, status.Byte1, status.Byte2);
            return (retu);
}
```